MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

Option : **Ingénierie des Systèmes d'Information**

THEME :

L'apprentissage profond avec les modèles de réseaux de neurones pour la détection des plaques d'immatriculations

Etudiant(e) : « **Kaouba Fatima Zohra**»

Encadrant : « **BENAMEUR ABDELKADER**»

Année Universitaire 2020-2021

REMERCIEMENTS

Je ne sais par quoi commencer, si ce n'est de remercier Dieu le tout puissant de m'avoir donné la force et le courage de continuer à fournir des efforts et d'arriver au point au quel je suis arrivés aujourd'hui. Je tiens tout d'abord à exprimer mes reconnaissance envers Les cadres de l'université Mr Ghazar le vice recteure et Mr Djebbara le Chef de d épartement , pour ses conseils, ses aides, son sympathie, ses contribution et pour le climat de travail agréable. Je tiens à exprimer mes plus sincères remerciements à mon encadreur Mr Benaamer pour la confiance, pour sa disponibilité, ses conseils judicieux et pour sa supervision éclairée tout au long de la rédaction du mémoire

. Enfin, je suis extrêmement reconnaissante à Mes responsables plus particulièrement Mr le Wali de la Wilaya Aissa Boulihia qui m'a donnée l'occasion pour continuer mes études supérieures, et a tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Résumé

Les systèmes ALPRs (Automatic License Plate Recognition) sont utilisées dans plusieurs domaines ces derniers ont recours à différentes techniques de détection des plaques d'immatriculation, et cela, tout en essaye de résulter les meilleures performances qui soient. L'une de ces techniques est l'utilisation du l'apprentissage profond grâce aux modèles de réseaux de neurones. Cela nous amène à s'interroger sur lequel de ces modèles est le plus adéquat pour réaliser le processus de détection des plaques d'immatriculations des camions à la rentrer des CETs « Centre d'enfouissement Techniques » afin de maîtriser le bon déroulement de dépôt des ordures aménageurs dans les casiers des mêmes CET à la wilaya de Mostaganem.

L'objectif de cette étude est de proposer une solution qui détermine les différentes approches basées sur les modèles de réseaux de neurones utilisées pour la détection des plaques d'immatriculation, et ce afin d'en tirer ceux qui sont les plus performants et les plus efficaces.

Pour répondre à la problématique, nous nous sommes mis à la recherche et à l'étude des différents travaux liées à la détection des plaques d'immatriculation. Par la suite, nous avons fait une synthèse de ce que nous avons pu trouver. En effet, les réponses récoltées montrent qu'il y a deux grandes familles de modèles de réseaux de neurones dédiées pour la détection d'objets, entre autres, les plaques d'immatriculation. La famille dite à une seule phase présente les meilleurs critères pour répondre à notre problématique, à savoir, quels sont les meilleurs modèles de réseaux de neurones utilisés pour la détection des plaques d'immatriculation. Cette constatation a été vérifiée à travers l'implémentation de deux modèles : RetinaNet et YOLOv3, deux modèles tirés de la famille à une phase, ils ont présenté des résultats très satisfaisante, surtout concernant le modèle YOLOv3.

Une modeste contribution a été proposée dans le but d'améliorer les taux de précision de la détection des plaques d'immatriculation. En effet, nous avons développé une application qui a été testée pour la détection et reconnaissance des plaques d'immatriculation des véhicules.

Table des matières

Table des figures

Introduction Générale

Contexte

Aujourd'hui, les systèmes ALPRs (Automatic License Plate Recognition) apportent une réelle valeur au besoin de détection et de reconnaissance des plaques d'immatriculation des véhicules. Ces systèmes ALPRs s'articulent sur diverses techniques et approches pour la réalisation de l'étape de détection de la plaque, mais aussi celle de la reconnaissance des caractères. Concernant l'étape de détection, celle-ci fait partie d'un plus grand domaine qui est la détection d'objets au sein des images et des vidéos. L'une des approches les plus connues dans ce domaine est l'apprentissage en profondeur (le deep learning). En effet, le deep learning bénéficie des modèles des réseaux de neurones artificielles qui sont connus pour leurs capacités d'apprentissage, de mémorisation, mais surtout de traitement des données incomplètes. D'après plusieurs travaux et réalisations scientifiques [1], les réseaux de neurones font preuve d'une efficacité remarquable en terme de précision et de temps d'exécution dans la détection des objets. Il serait pertinent de voir l'impact de l'application des modèles de réseaux des neurones dans la détection des plaques d'immatriculation.

L'objectif de cette étude est de déterminer les différentes approches basées sur les modèles de réseaux de neurones utilisées pour la détection des plaques d'immatriculation, et ce afin d'entirer ceux qui sont les plus performants et les plus efficaces pour implémenter notre système.

Problématique

Dans l'optique d'utiliser l'apprentissage profond pour le traitement et la localisation des plaques d'immatriculation des camions aménageurs a la rentré des CETs, il y a lieu de savoir qu'il existe plusieurs méthodes et modèles de détection qui présentent des performances plus ou moins différentes, chacun, a été proposé à destination d'un contexte de détection d'objets bien particulier. Cela nous amène à s'interroger lequel d'entre tous ces modèles est le plus aptes à être utilisé pour la détection des plaques d'immatriculation des camions.

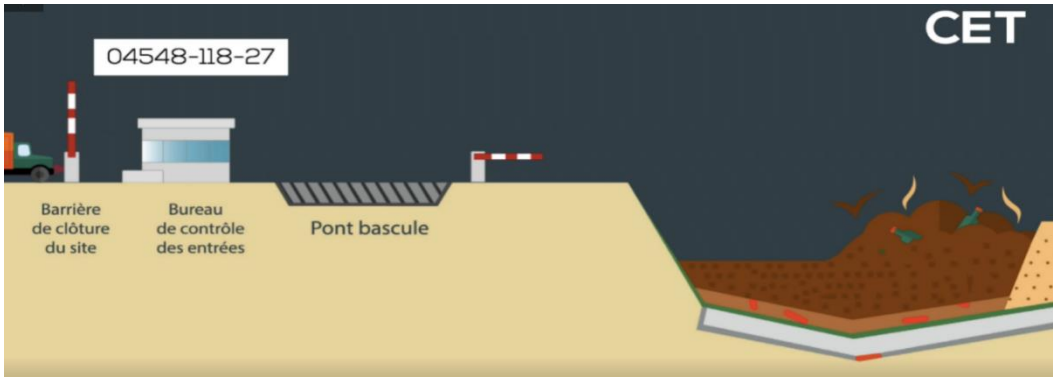


Figure I.1 un schéma représente l'objectif de notre système à la rentrer des CETs

Objectif

Dans la tentative de donner une réponse à la problématique, plusieurs études et recherches doivent être faites dans le cadre de l'utilisation de l'apprentissage profond plus précisément des modèles de réseaux de neurones pour la détection des plaques d'immatriculation. Nous voudrions départager entre les différents modèles existants, et ce à travers une analyse de leurs performances dans le processus de détection. Nous nous engageons à étudier la réelle portée de l'utilisation des modèles des réseaux de neurones dans la détection des plaques d'immatriculation, et ce afin de retourner le plus à même à être utilisé pour répondre aux critères de choix dans le processus de détection.

Organisation

Ce mémoire suivra les points suivants :

Chapitre I : représente l'ensemble des notions de base liées au problème de détection d'objets, et voir les concepts relatifs au deep learning et aux réseaux de neurones, ainsi qu'une introduction aux systèmes de reconnaissance des plaques d'immatriculation.

Chapitre II : englobe les différentes approches deep learning dans le domaine de la détection des plaques d'immatriculation et un état de l'art sur les travaux de recherche, une étude comparative entre les différents types d'approches plus une recherche bibliographique sera menée afin de mettre en évidence les différentes techniques qui sont basées sur les modèles de réseaux de neurones, et qui sont utilisées pour la détection des plaques d'immatriculation.

Chapitre III : une contribution sera proposée ayant pour but d'améliorer la précision du processus de détections des plaques d'immatriculations

Chapitre IV : implémentation d'une application au niveau des **C**entre d'**E**nfouissement **T**echnique CET ainsi qu'une synthèse sur les pistes de travaux futurs de ce domaine de recherche.

Et pour terminer, une conclusion générale par rapport à l'ensemble de ce qui va être présenté durant ce travail.

CHAPITRE I : Concepts et généralités

I.1 Introduction

En vue de constater l'application du l'apprentissage profond « Deep Learning » sur la détection des plaques d'immatriculation, il nous faut tout d'abord définir les notions importantes liées à cette objectif. En effet, le problème des détections des plaques d'immatriculation est un cas particulier du problème de détection d'objets dans les images. Ce dernier est généralement abordé à travers la fenêtre du machine learning (l'apprentissage automatique), où il y a toute une panoplie de techniques et d'approches, notamment, l'utilisation du Deep Learning avec les réseaux de neurones.

Ce chapitre contiendra les points suivants :

- **Section I.2** : L'introduction du problème de détection d'objets
- **Section I.3** : La définition du machine learning.
- **Section I.4** : généralités sur l'apprentissage profond
- **Section I.5** : Le deep learning et les réseaux de neurones.
- **Section I.6** : Les principaux concepts liés à l'apprentissage des réseaux de neurones.
- **Section I.7** : La conclusion de ce chapitre.

I.2 Détection d'objets dans une image

La détection des objets dans une image est l'un des principaux axes dans le domaine de la vision par ordinateur. Elle a pour objectif la localisation et la reconnaissance des différentes classes d'objets au sein de l'image. Ce procédé passe, en général par plusieurs étapes de traitement d'image tel que : le filtre gaussien, la convolution, etc. Et cela, afin de générer des sorties sous forme de coordonnées des rectangles, aussi appeler les boites englobantes, qui délimitent les objets détectés au sein de l'image.

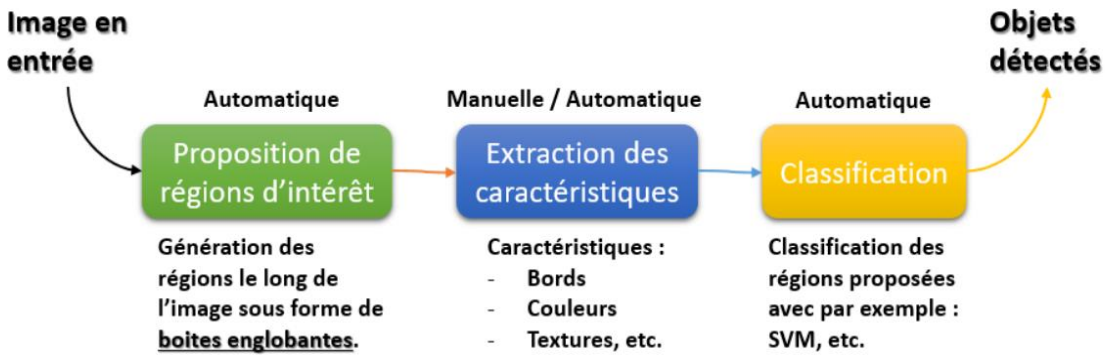


FIGURE I.2 – Processus général de détection d'objets.

La figure I.2 représente le processus général de détection d'objets dans une image. Ce processus est caractérisé par trois grandes étapes :

1- **La proposition des régions d'intérêt** : cette étape consiste à la division de l'image d'entrée en plusieurs régions susceptibles de contenir des objets. Son implémentation est très variée, elle diffère d'une technologie à une autre. Par contre, son exécution est toujours automatique, que ce soit par des approches de machine learning classiques, ou par le deep learning.
2- **L'extraction des caractéristiques** : chaque région de l'image fera l'objet d'une extraction de ses caractéristiques intrinsèques comme : la couleur, la texture, les formes, les bords, etc. Ces caractéristiques seront nécessaires pour l'étape de classification et donc de détection. Cette étape d'extraction des caractéristiques peut être soit automatique par le biais des réseaux de neurones par exemple, ou manuelle par le biais d'un expert humain dans le cas des approches de machine learning plus classiques.
3- **La classification** : après avoir extrait les caractéristiques intrinsèques d'une région, cette dernière fera l'objet d'une classification afin de prédire la nature de l'objet se trouvant à l'intérieur. La sortie sera une boite englobante l'objet en question avec un score de confiance qui représente à quel point la détection est-elle sûre. L'étape de classification est une étape automatique qui peut être exécutée avec un classifieur SVM par exemple.

Aujourd'hui, la détection d'objets est souvent abordée à travers la fenêtre du machine learning. Il existe beaucoup de techniques et d'implémentations dans ce domaine, chacune, présentant ses avantages et ses inconvénients. Toujours dans le contexte de la détection d'objets, nous verrons dans les sections suivantes quelques notions générales autour du machine learning.

I.3 Le machine learning

Le machine learning (ou apprentissage automatique) [2] est un sous domaine de l'informatique qui s'intéresse à la construction d'algorithmes s'appuyant sur un ensemble de données d'un phénomène donné, et ce afin de pouvoir résoudre certains problèmes pratiques.

Le machine learning peut être aussi défini comme le processus suivant :

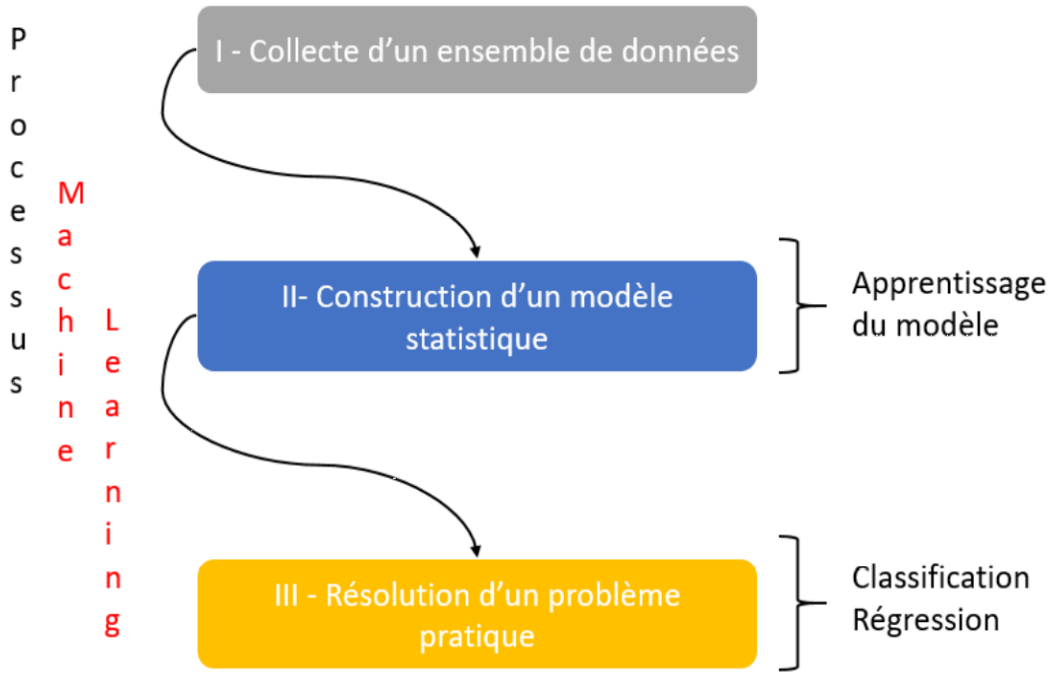


FIGURE I.3 – Processus Machine Learning.
Le processus de machine learning est caractérisé par ces trois étapes.

La première étape qui est la collecte d'un ensemble de données, elle peut se faire soit par l'homme, soit être générée par un algorithme. Cet ensemble constituera l'exemple d'apprentissage d'un modèle statistique pour un phénomène donné.

La deuxième étape consiste à la construction d'un modèle statistique sur la base d'un ensemble de données. Cette construction se fera à l'aide d'algorithmes de machine learning qui feront subir un apprentissage au modèle sur l'ensemble de données.

La dernière étape, la résolution d'un problème pratique est le traitement de nouvelles instances de données grâce au modèle obtenu dans l'étape précédente. Ces traitements consistent en :

- La classification : est le fait d'assigner automatiquement une étiquette à un exemple non étiqueté. Exemple : la détection de spam.

- La régression : est la prédiction de la valeur réelle de l'étiquette pour un exemple non étiqueté. Exemple : l'estimation du prix d'un domicile suivant sa localisation, sa surface, le nombre de chambre. Les algorithmes de machine learning ont donc pour objectif de modéliser les différents phénomènes afin de pouvoir traiter les nouvelles observations et ainsi résoudre les problèmes liés à ces phénomènes.

I.4 le Deep Learning

I.4.1 Définition

L'apprentissage profond « le deep Learning » est un sous-domaine de l'apprentissage automatique traitant d'algorithmes inspirés par la structure et la fonction du cerveau appelés réseaux de neurones artificiels. Pour ainsi dire, Il copie le fonctionnement de nos cerveaux. Les algorithmes d'apprentissage profond sont comme l'organisation des systèmes sensoriels ou` chaque neurone s'associe et transmet des informations. Les modèles d'apprentissage profond fonctionnent en couches et un modèle typique comporte au moins trois couches. Chaque couche accepte les informations de la précédente et les transmet à la suivante.

I.4.2 L'histoire : **naissance, d´déclin et prospérité**

Les modèles profonds peuvent être appelés réseaux de neurones avec des structures profondes. L'histoire des réseaux de neurones peut remonter aux années 40 [3], et l'intention initiale était de simuler le système cérébral humain pour résoudre des problèmes d'apprentissage généraux d'une manière fondée sur des principes. Il était populaire dans les années 1980 et 1990 avec la proposition d'un algorithme de rétro propagation par Hinton et al [4]. Cependant, en raison du sur-ajustement de l'entrainement, du manque de données d'entrainement à grande échelle, de la puissance de calcul limitée et de l'insignifiance des performances par rapport aux autres outils d'apprentissage automatique, les réseaux de neurones sont tombés en d´désuétude au début des années 2000.

L'apprentissage profond est devenu populaire depuis 2006 avec une percée dans la reconnaissance vocale [5]. Le rétablissement de l'apprentissage profond peut être attribué aux facteurs suivants :

- L'émergence de données d'entrainement annotées à grande échelle, comme ImageNet [6], pour montrer pleinement sa très grande capacité d'apprentissage ;
- Développement rapide de systèmes informatiques parallèles hautes performances, tels que les processeurs graphiques GPU ;
- Avancées significatives dans la conception de structures de réseaux et de stratégies de l'entrainement. Avec un pré-entrainement non supervisée, une bonne initialisation est fournie. Avec l'augmentation des données, le problème de sur-apprentissage dans l'entrainement a été soulagé [6]. Avec la normalisation par lots, l'entraînement du réseau de neurones très profond devient assez efficace [7]. Pendant ce temps, diverses structures de réseau, comme AlexNet [6], Overfeat [8], GoogLeNet [9], VGG [10] et ResNet [11], ont été largement étudiées pour améliorer les performances.

I.5 Machine Learning et Deep Learning

Il faut savoir que le machine learning est un vaste domaine comportant plusieurs aspects et philosophie d'algorithmes et d'implémentations. Le deep learning étant un sous domaine du machine learning, ce dernier peut être divisé en deux catégories :

- Le machine learning classique : il s'agit de l'intégral des méthodes et algorithmes qui ne sont pas basés sur le deep learning, exemple : KNN, Clustering, etc.
- Le deep learning : il s'agit des algorithmes et des approches qui sont basés sur le deep learning, exemple : les réseaux de neurones.

Là où **le machine learning** classique peut ne pas donner de très bon résultats, **le deep learning** vient pour essayer de palier à ce problème, et ainsi augmenter de façon significative la précision des résultats.

Ce qui fait la force du deep learning c'est qu'il est caractérisé par des architectures complexes, articulées sur plusieurs transformations non-linéaire, ce qui lui permet de mieux appréhender le phénomène étudié par rapport au machine learning classique, et ainsi mieux le modéliser.

L'une des différences entre le modèle d'apprentissage automatique (Machine Learning) et le modèle d'apprentissage profond (Deep learning) réside dans la zone d'extraction des descripteurs. L'extraction des descripteurs est effectuée par l'humain dans l'apprentissage automatique, tandis que le modèle d'apprentissage en profondeur se comprend lui-même.

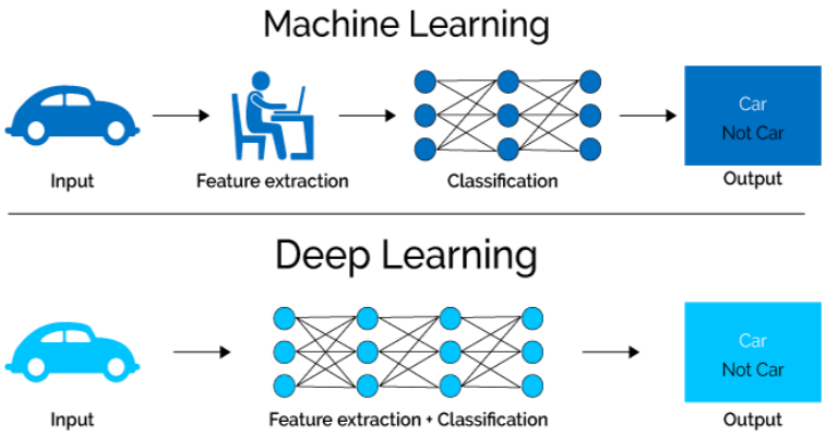


FIGURE I.4 – Machine Learning vs Deep Learning.

La figure I.4 montre la différence qu'il y a entre un processus **machine learning** classique et un processus de **deep learning** sur un exemple de détection de l'objet voiture au sein d'une image en entrée.

L'étape d'extraction des caractéristiques est l'extraction d'un vecteur de caractéristiques pour chaque région proposée. Ces caractéristiques peuvent être : des bords, des textures, des couleurs,

des formes, etc. Elles sont déterminantes pour l'étape de classification pour la détection de l'objet : voiture.

Pour un processus de machine learning classique, l'extraction des caractéristiques est manuelle, elle se fait grâce à un expert humain responsable de la description explicite des caractéristiques de l'objet recherché : la voiture. Ces caractéristiques peuvent être les formes, les couleurs ou les textures de la voiture.

Quant-au cas du deep learning, l'extraction des caractéristiques se fait automatiquement, il n'y a pas besoin de les exprimer explicitement. En effet, l'architecture sous plusieurs niveaux du deep learning est capable d'extraire automatiquement les différentes caractéristiques des objets. Par contre, cela requière une plus grande quantité de données d'apprentissage que pour le machine learning classique.

Le paradigme du deep learning est représenté par les réseaux de neurones artificielles inspirés les réseaux de neurones biologiques du cerveau. Les algorithmes du deep learning sont à base des réseaux de neurones. Ils sont donc capables d'appréhender les phénomènes les plus difficiles grâce à l'architecture complexe que leur confèrent les réseaux de neurones.

I.6 Le Deep Learning et les réseaux de neurones

Dans cette section nous allons voir les réseaux de neurones qui sont une implémentation directe du Deep learning.

I.6.1 Les réseaux de neurones

Les réseaux de neurones artificiels sont une inspiration des réseaux de neurones du cerveau. Ils essayent de reproduire le fonctionnement du cerveau et ainsi avoir les mêmes fonctions que ce dernier. Les réseaux de neurones ont la particularité de pourvoir approximer n'importe quelle fonction, et donc n'importe quel phénomène. Ce qui les rend extrêmement efficaces dans la résolution des problèmes pratiques.

Les réseaux de neurones artificielles sont en général constitués de plusieurs couches de neurones comme l'illustre la figure I.5 :

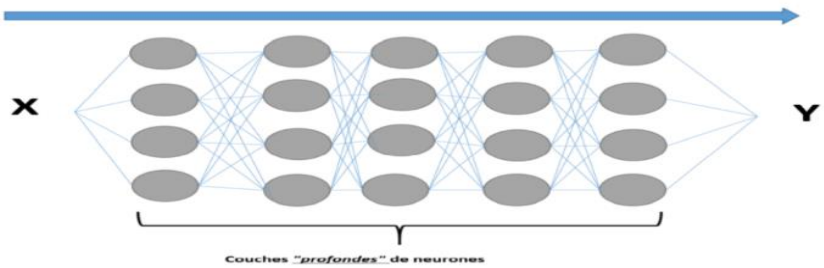


FIGURE I.5 – Illustration d'un réseau de neurones.
Les couches au milieu sont dites couches profondes ou couches cachées.

La figure I.5 montre une architecture d'un réseau de neurones. Nous pouvons voir que ce réseau de neurones est constitué de plusieurs couches de neurones. Ces couches de neurones peuvent comporter plusieurs milliers, voire plusieurs millions de neurones, et c'est cela qui définit la complexité d'un réseau de neurones.

Le réseau de neurones tel qu'il est, doit subir un entraînement en envoyant une grande quantité d'éléments X à tester pour rendre les neurones autonomes : c'est la phase d'apprentissage. Après la phase d'apprentissage, un nouvel élément X pourra être traité convenablement afin d'extraire automatiquement les caractéristiques intrinsèques de cet élément, et ainsi pouvoir donner un résultat Y. Le X peut avoir plusieurs formes, il peut être soit une image, une vidéo, un son, du texte, etc

La constitution élémentaire d'un réseau de neurones est le neurone. La figure I.6 montre la structure d'un neurone artificiel, aussi appelé perceptron.

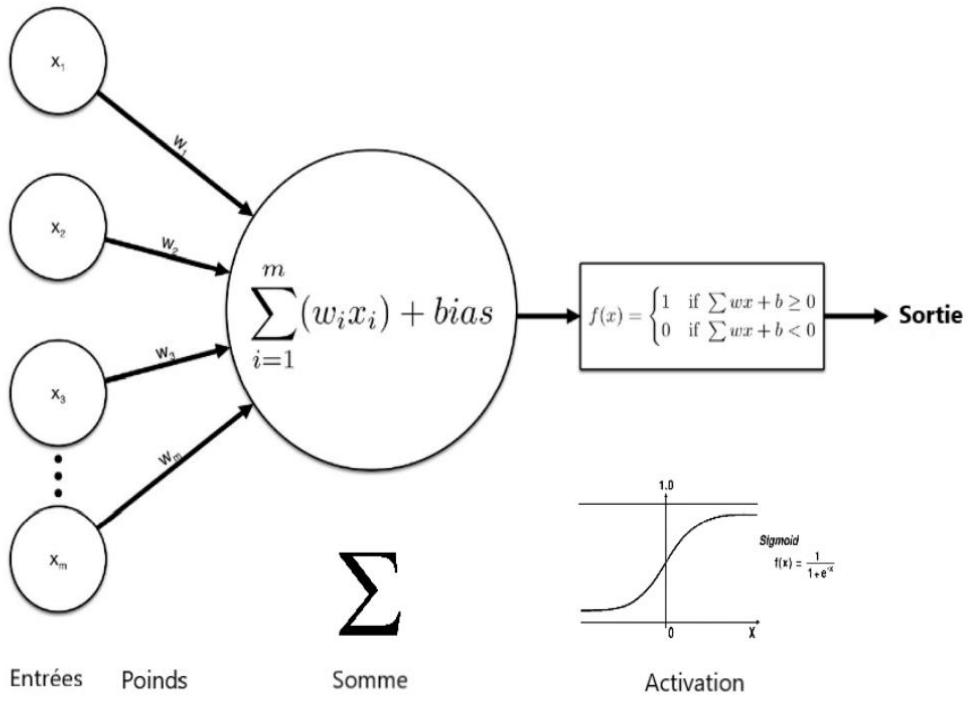


FIGURE I.6 – Structure d'un neurone artificiel.
Aussi appelé perceptron.

Le perceptron a été introduit par Rosenblatt en 1958 [12]. Il constitue le réseau de neurones le plus simple qui soit.

La figure I.6 montre la structure d'un perceptron qui est composé de :

- **Entrées :** Sont des neurones, généralement d'une couche antérieure, qui communiquent de l'information x.
- **Poids :** Les entrées x sont multipliées par les poids w, ils représentent les paramètres internes du perceptron.
- **Somme** : Toutes les multiplications sont sommées avec ajout d'un biais.

- **Activation** : Le résultat de la sommation est donné à une fonction d'activation, ici comme exemple la fonction sigmoid.
- **Sortie** : La sortie sera donc le résultat de la fonction d'activation dans l'intervalle [0 ;1].

I.7 Apprentissage d'un réseau de neurones

I.7.1 Apprentissage : explication

Depuis la figure I.6 vu ci-dessus, un point est à préciser concernant les poids w. En effet, durant la phase d'apprentissage d'un réseau de neurones, l'objectif sera d'évaluer et de trouver les valeurs optimales de ces poids. Ils représentent donc les paramètres internes du réseau de neurones.

Il existe aussi d'autres paramètres pour le réseau de neurones appelés : les **hyperparamètres.** Ce sont des paramètres qui doivent être pré-défini dès le début pour cause qu'ils régissent l'évolution de la phase d'apprentissage et conditionnent la précision des valeurs des poids à estimer.

La détermination des valeurs des hyper paramètres est souvent liée à un problème d'optimisation en vue de trouver les meilleures valeurs qui soient, et cela, afin d'avoir le meilleur apprentissage qui donnerait les meilleures valeurs des poids (les paramètres internes) du réseau de neurones.

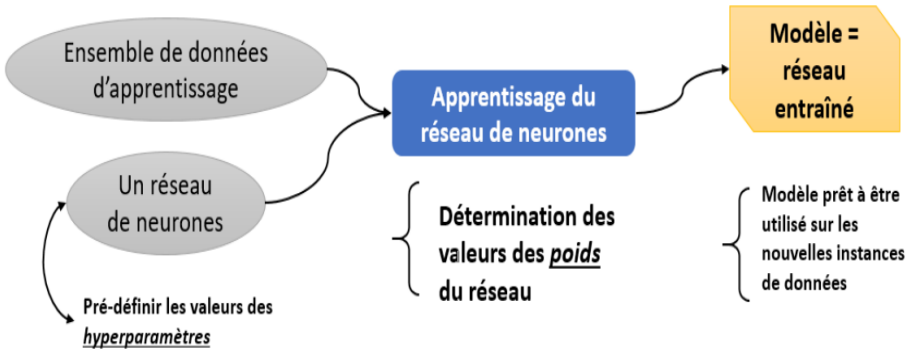


FIGURE I.7 – Schéma d'apprentissage d'un réseau de neurones.

La figure I.7 montre le processus général de la phase d'apprentissage d'un réseau de neurones. L'objectif étant de trouver les meilleures valeurs des poids du réseau, afin d'avoir un modèle performant.

L'apprentissage est **conditionné** par l'introduction des valeurs des hyperparamètres. Un réseau de neurones peut avoir plusieurs hyperparamètres, parmi ceux les plus connus :

- Epoch : signifie le nombre de fois où les données d'entraînement vont être passées à travers le réseau de neurones. Il peut avoir comme valeur : 5, 15, 25, 100, 200.
- Batch : signifie le nombre des échantillons à faire passer dans le réseau de neurones avant la mise à jour des poids (les paramètres internes du réseau). Il peut prendre comme valeur : 4, 6, 8, 16, 32.
- Iteration : signifie le traitement d'un lot d'images à travers le réseau. Ce lot est égal au batch. Iteration est généralement égale au nombre de données d'entraînement divisé sur le batch.
- Learning rate : ou taux d'apprentissage, ou même pas d'apprentissage, il signifie le degré d'ajustement à appliquer durant la mise à jour des poids du réseau de neurones. Il peut prendre comme valeur : 0.1, 0.01, 0.001, 0.0001.
- Backbone : il s'agit d'un sous réseau de neurones responsable de l'extraction des caractéristiques intrinscèques de l'image, et cela, dans le but d'effecteur le processus de détection. Exemples de bachbone : Darknet, VGG, Resnet, etc.

L'apprentissage d'un réseau de neurones est basé sur deux points essentiels :

- Le calcul de l'erreur de prédiction : ce point est assuré par une fonction appelée : fonction de perte. Cette fonction calcule une valeur qui représente l'erreur de prédiction des instances de données d'entraînement passées à travers le réseau de neurones. L'erreur quadratique [13] et l'entropie croisée [14] sont des exemples de fonction de pertes.
- L'ajustement des valeurs des poids du réseau : ce point est assuré par un algorithme d'optimisation. Il est responsable d'ajuster les valeurs des poids du réseau de neurones, et ce à partir de l'erreur de prédiction générée par la fonction de perte du réseau de neurones. Exmple d'algorithmes d'optimisation : Descente de gradient [15], Adam [16], etc. La figure I.7 résume le processus d'apprentissage d'un réseau de neurones, et ce à travers un cycle désignant le rôle de la fonction de perte et de l'algorithme d'optimisation.

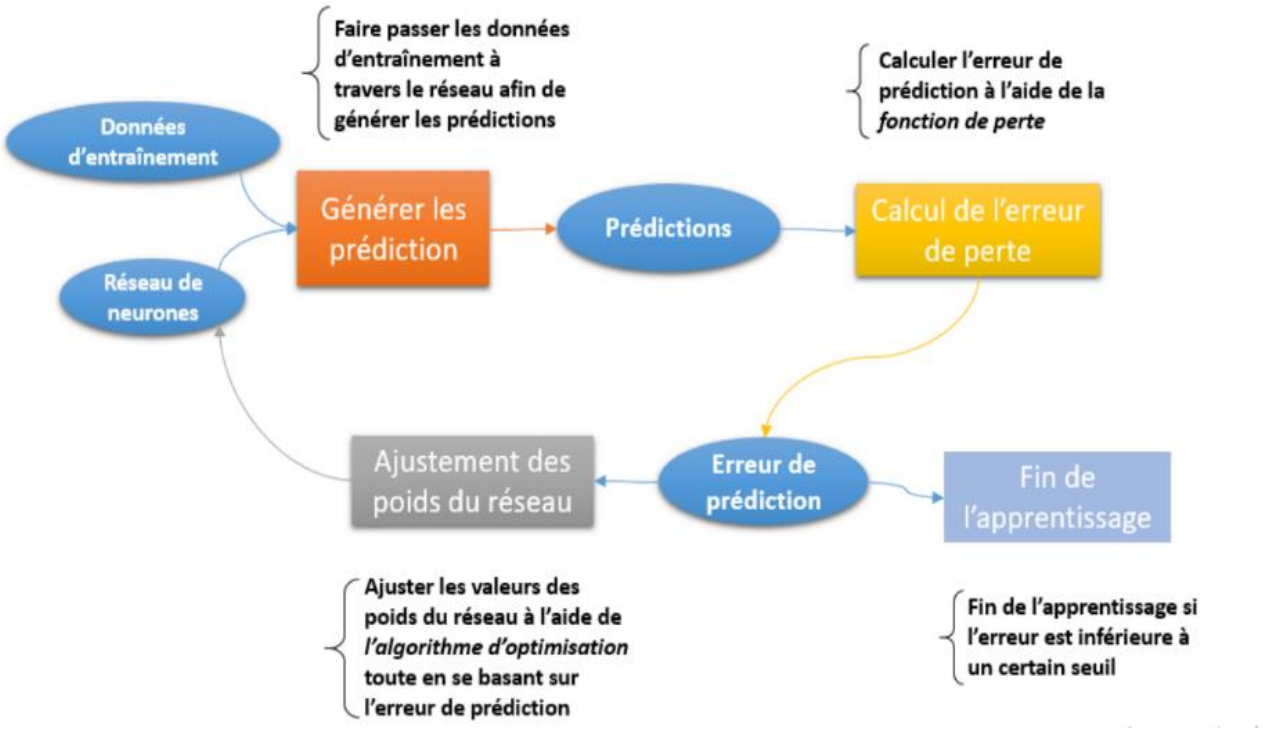


FIGURE I.8 – Cycle d'apprentissage d'un réseau de neurones.

I.7.2 Méthodes d'apprentissage d'un réseau de neurones

Comme nous l'avons vu dans la section précédente I.5, l'apprentissage d'un réseau de neurones consiste en l'évaluation des meilleures valeurs possibles des poids du réseau.

Initialement, les poids du réseau de neurones ont une valeur qui peut être, soit aléatoire, soit définie au préalable. Une valeur initiale qui est aléatoire pour les poids du réseau a pour conséquence un temps d'apprentissage très lent. C'est pour cela, que dans la pratique, cette approche n'est pas conseillée. Il en découle une autre approche qui est d'initialiser les poids du réseau de neurones avec des valeurs bien spécifiques.

Nous pouvons distinguer deux méthodes d'apprentissage, suivant si les valeurs initiales des poids sont aléatoires ou non.

I - Méthode sans transfert : Ici, les valeurs des poids du réseau de neurones sont initialisées aléatoirement. Le temps d'apprentissage peut s'avérer très lent pour cause que la recherche des valeurs optimaux des poids peut prendre beaucoup de temps.

Cette approche est considérée comme non pratique du fait des ressources qu'il faudrait déployer pour sa mise en œuvre.

II - Méthode avec transfert : L'apprentissage avec transfert est une méthode d'apprentissage automatique dans laquelle un modèle formé pour une tâche, similaire à la nôtre, est réutilisé comme point de départ pour un modèle sur une deuxième tâche.

En effet, les valeurs des poids du réseau de neurones, ou du modèle, ont initialement des valeurs bien spécifiques. Ces valeurs proviennent d'un apprentissage antérieur qui a été déjà fait avec un certain dataset.

L'avantage pour cette méthode, c'est que le fait d'avoir déjà fait un apprentissage, permet au nouvel apprentissage de se terminer beaucoup plus rapidement que lorsqu'il n'y avait un apprentissage antérieur.

Cela permet non seulement d'avoir un temps d'apprentissage relativement court, mais aussi, de réduire les ressources nécessaires pour l'apprentissage dont les données d'entraînement.

Comme exemple : dans un problème de détection d'objet dans une image, il serait plus judicieux d'utiliser un modèle pré-entraîné sur un ensemble d'image vaste et volumineux tel que l'ensemble ImageNet [17]. Cela nous permettra de former notre modèle en peu de temps et sur la base d'un ensemble de données réduit. Ces modèles pré-entraînés peuvent prendre des jours ou. des semaines pour s'entraîner sur du matériel moderne.

À noter que, mise à part le fait que l'apprentissage du modèle soit plus court, il n'est, souvent, pas évident de démontrer un réel avantage au niveau des performances du modèle quand nous utilisons l'apprentissage avec transfert. En effet, il faudrait dans ce cas-là, passer par une étape d'évaluation de ce modèle.

Selon Lisa Torrey et Jude Shavlik [18], trois avantages possibles sont en faveur de l'apprentissage avec transfert :

1. Initialisation plus élevée : le point du départ de l'apprentissage présente déjà des compétences meilleures.
2. Pente de convergence plus élevée : le taux d'amélioration des compétences est plus élevé.
3. Asymptote supérieure : la convergence du modèle est meilleure.

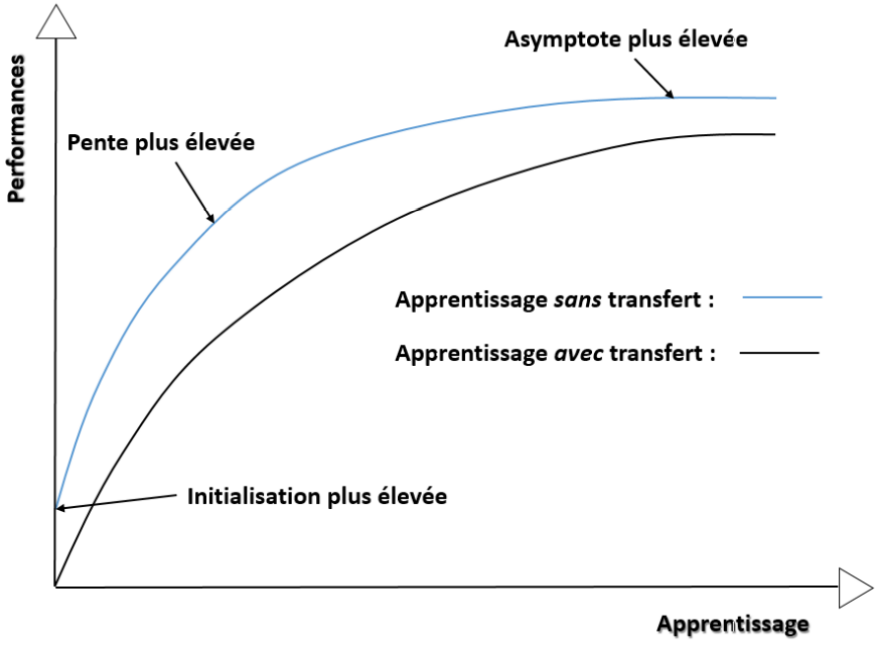


FIGURE I.9 – Apprentissage sans vs avec transfert

La figure I.9 montre les différences potentielles entre l'apprentissage sans et avec transfert.

I.8 Conclusion

Dans ce chapitre nous avons pu voir les principales notions autour du domaine de détection d'objets, de la machine Learning, mais surtout autour des réseaux de neurones. Ces derniers ont la réputation d'être très efficaces dans la modélisation de problèmes complexes, entre autres, la détection d'objets dans les images.

À la lumière de ce qui a été vu, il serait très pertinent d'utiliser les réseaux de neurones pour la détection des plaques d'immatriculation. Dans le chapitre suivant, nous présenterons un état de l'art autour de cette optique.

Chapitre II État de l'art

II.1 Introduction

Étant donné le cadre de recherche de ce mémoire, à savoir, l'application du deep learning dans la détection des plaques d'immatriculation des camions aménager, ce chapitre fera l'objet d'un état de l'art autour de l'utilisation des modèles des réseaux de neurones pour la détection des plaques d'immatriculation.

Durant nos recherches, nous avons trouvé plusieurs travaux faisant sujet de plusieurs méthodes destinées à la détection des plaques d'immatriculation. Nous avons pu noter que ces travaux pouvaient être classés suivant deux familles :

- Ceux qui sont basés sur la détection de la plaque sur deux phases.
- Et ceux qui sont basés sur la détection de la plaque sur une seule phase.

Chacune de ces familles détient tout un éventail de techniques qui sont axées sur les modèles de réseau de neurones, mais qui ont tous le même objectif qui est la réalisation du processus de détection.

Ce chapitre est divisé comme suit :

- **Section II.2** : Présentation de quelques critères d'évaluation des performances des modèles de réseau de neurones dans le problème de détection des plaques d'immatriculation.
- **Section II.3** : Présentation de la famille de détection à deux phases avec les travaux correspondants.
- **Section II.4** : Présentation de la famille de détection à une seule phase avec les travaux correspondants.
- **Section II.5** : Comparaison des deux familles en énumérant les points forts et les points faibles de chacune, le tout, accompagné d'une synthèse.
- **Section II.6** : Une étude comparatives entre deux modèles tirés de la famille à une seule phase.
- **Section II.7** : La conclusion de ce chapitre.

II.2 Critères d'évaluation des réseaux de neurones

Dans cette section nous allons voir quelques différents critères d'évaluation des modèles de réseau de neurones. Ces critères nous aiderons à mieux comprendre les travaux que nous allons présenter dans les sections suivantes.

Un modèle de réseau de neurones subi un apprentissage sur un ensemble de données d'entraînement, par la suite, ce modèle doit être testé sur un ensemble de données de test afin d'être

évalué et jugé de ses performances et de sa capacité de généralisation. La généralisation d'un modèle est sa capacité à effectuer des prédictions robustes sur des nouvelles instances de données. Dans le contexte du problème de détection d'objet, notamment, celui de la détection des plaques d'immatriculation, nous pouvons trouver dans la littérature plusieurs critères de performances des modèles de réseau de neurones.

II.2.1 La Précision et le Rappel

Ces deux métriques sont souvent utilisées pour présenter les performances des modèles pour la détection d'objets. Elles s'énoncent comme suit :

p : Précision

$$p = \frac{VP}{VP+FP} = \frac{VP}{Total\ des\ resultats\ positifs}$$

r : Rappel

$$r = \frac{VP}{VP+FP} = \frac{VP}{Total\ cas\ reels}$$

Où : V P = Vrai Positif, F P = Faux Positif, et F N = Faux Négatif

Un VP est généralement calculé à travers la métrique **IOU (Intersection Over Union).**

II.2.2 IOU (Intersection Over Union)

Cette métrique prend en entrée deux paramètres :

- Les coordonnées estimées d'un objet détecté au sein de l'image.
- Les vraies coordonnées de ce même objet tirées du label accompagnant l'image.

Ces coordonnées sont souvent représentées par un rectangle

Figure II.1 Entrées de la métrique IOU

L'IOU retourne une valeur entre 0 et 1, et qui symbolise l'exactitude de la détection de l'objet. La figure montre l'équation utilisée pour le calcul de l'IO

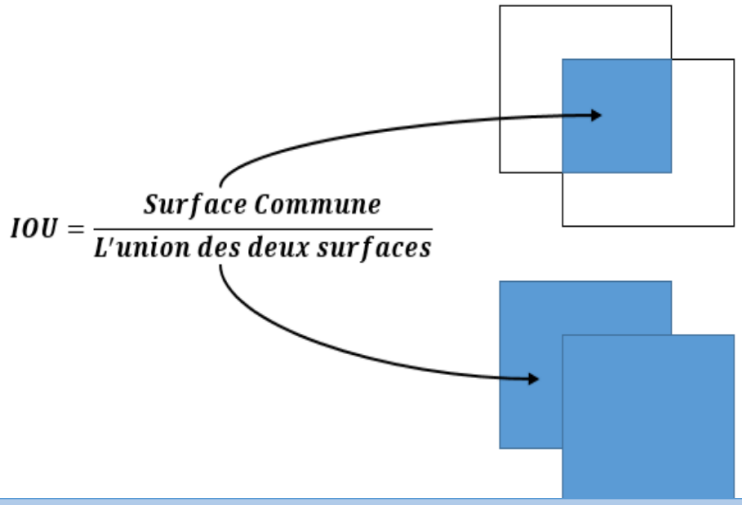


Figure II.2 Calcule de l'IOU
Un carré représente les coordonnées de l'objet au sein de l'image

Au-delà d'un certain **seuil** pour la métrique **IOU**, un **VP** est validé, sinon, il est considéré comme un **FP**.

Les **FNs** sont tout simplement les objets qui n'ont pas été détectés.

Exemple : pour la détection d'une plaque d'immatriculation, le fait de ne pas détecter une certaine plaque au sein de l'image, celle plaque sera considérée comme un **FN.**

L'algorithme 1 résume le processus de calcul d'un **VP** et d'un **FP** avec la métrique **IOU**

Algorithme 1 Algorithme de calcul d'un VP et d'un FP avec l'IOU

Entrée : Coordonnées estimées tirées de la détection d'un objet + vrais coordonnées de l'objet en question.
Sortie : L'objet détecté est soit un VP ou un FP.

```
1: procédure GET_VP_FN
2:     coord1 = coordonnées estimées de l'objet détecté.
3:     coord2 = les vrais coordonnées de l'objet en question.
4:     IOU = getIOU(coord1, coord2).
5:     si IOU ¡ seuil_IOU alors
6:         L'objet détecté est considéré comme un VP.
7:     sinon
8:         L'objet détecté est considéré comme un FP.
9:     fin si
10: fin procédure
```

getIOU(coord1, coord2) : une fonction qui retourne la métrique IOU entre coord1 et coord2.
seuil_IOU : est défini par l'utilisateur.

II.2.3 Le F-Score

Le F-Score est lui aussi très utilisé dans l'évaluation des performances des modèles de réseau de neurones pour la détection d'objets. Il s'énonce comme suit :

$$F - Score = \ 2 \times \frac{Precision \times Rappel}{Precision + Rappel}$$

II.3 Famille de deux phase

Cette famille fait référence aux réseaux de neurones convolutifs à proposition de régions comme : **RCNN, Fast-RCNN**, ou **Faster-RCNN**.
Elle a la particularité d'avoir un pipeline de détection composé de deux grandes étapes :

1. Proposition de régions : Cette étape fait en sorte de proposer des régions candidates pouvant contenir des éventuelles objets, et cela, en divisant l'image d'entrée en de petites zones. Elle se base sur des techniques de proposition de régions tel que la recherche sélective, ou encore, l'utilisation d'un ou de plusieurs réseaux de neurones spécialement dédiés pour cette tâche.
2. Extraction des caractéristiques et Classification : Chaque région proposée par l'étape précédente subi une extraction des caractéristiques par le billet d'un CNN, suivi d'une classification avec, éventuellement, un SVM (Support Machine Vector), ou un modèle de classification.
La figure II.3 suivante schématise le processus général de détection à l'aide de la famille à deux phase

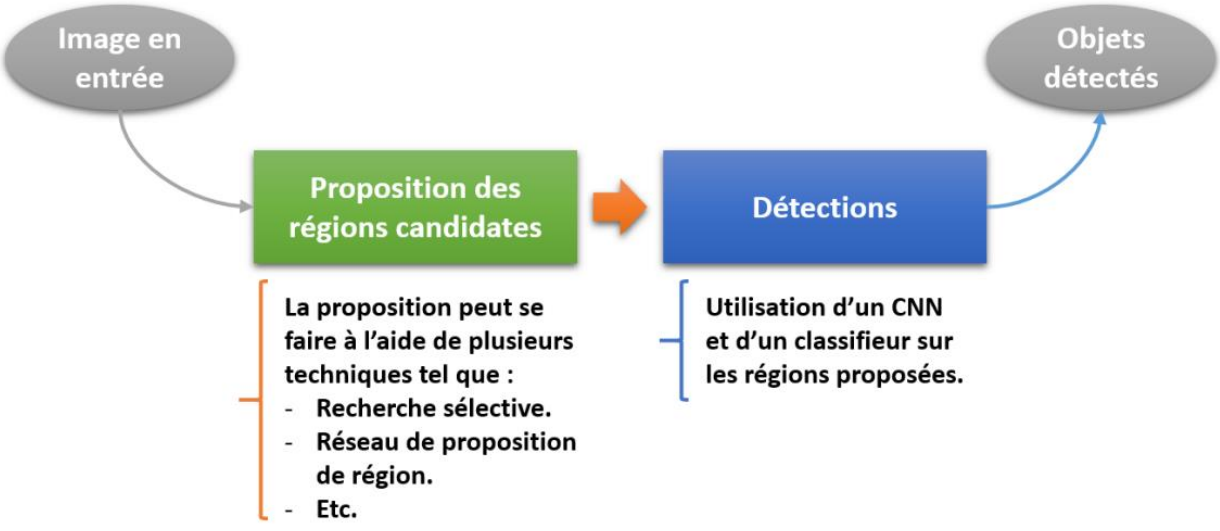


FIGURE II.3 – Processus de détection en deux phases

Nous présentons par la suite quelques-uns des modèles phares tirés de cette famille, accompagnés de quelques travaux utilisant ces modèles pour la détection des plaques d'immatriculation.

II.3.1 RCNN **[19]**

Le détecteur R-CNN se compose de quatre modules. Le premier module génère des propositions de régions indépendantes des catégories. Le deuxième module extrait un vecteur de descripteur de longueur fixe de chaque proposition de région. Le troisième module est un ensemble de SVM linéaires spécifiques à une classe pour classer les objets dans une image. Le dernier module est un redresseur de cadre englobant pour une prédiction de cadre englobant précis.

Le **RCNN** est un modèle de réseau de neurones à propositions de régions. La particularité de ce modèle est son utilisation de la recherche sélective pour l'étape de proposition des régions candidates. En effet, cette recherche peut se résumer comme suit :

1. À partir de l'image d'entrée, une segmentation initiale est effectuée pour générer un nombre assez important de zones.
2. Combiner récursivement ces zones suivant des critères de similarités afin d'engendrer des zones plus grandes.
3. Utiliser les zones générées pour proposer des régions candidates. Ces régions seront au nombre de 2000. À partir de ces 2000 régions candidates, une extraction des caractéristiques et une classification avec un SVM sont opérées afin de déterminer la localisation des objets recherchés.

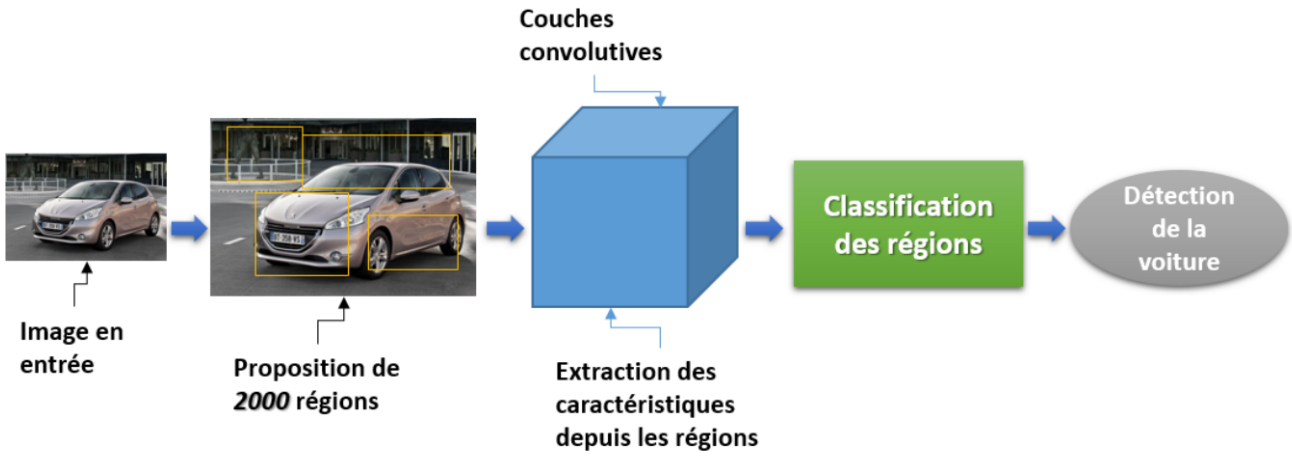


Figure II.4 Schéma général de détection avec R-CNN

Dans une comparaison a été faite entre la technique des fenêtres coulissantes et le R-CNN, ce qui a permis de conclure de façon ´évidente que le **R-CNN** est meilleur. Mais R-CNN est en soi lent et ne convient pas pour une utilisation dans des applications en temps réel. De plus, l'ensemble de données qu'ils ont utilisé a été pris de la caméra a une seule porte, laissant l'ensemble de données sans aucune variation.

Problèmes avec R-CNN :

- Il faut encore énormément de temps pour entrainer le réseau, car il faudrait classer 2000 propositions de région par image.
- Il ne peut pas être implémenté en temps réel car il faut environ 47 secondes pour chaque image de test.
- L'algorithme de recherche sélective est un algorithme fixe. Par conséquent, aucun apprentissage ne se produit à ce stade. Cela pourrait conduire à la génération de mauvaises propositions de régions candidates.

II.3.2 FAST R-CNN **[20]**

Contrairement à RCNN, Faster-RCNN utilise un réseau de proposition de région (Region Proposal Network - RPN) à la place de la recherche sélective pour la proposition des régions candidates permettant ainsi, d'accélérer le processus de proposition de façon très significative. Nous pouvons résumer le processus de détection du modèle Faster-RCNN comme suit :

1. À partir d'une image d'entrée, un réseau de neurones convolutif est utilisé pour extraire la carte de caractéristique de l'image.
2. Le RPN utilise cette carte de caractéristique pour prédire les régions.
3. Les régions proposées sont ensuite redimensionnées avec une opération de pooling, et cela, afin de pouvoir faire la classification et la régression des coordonnées des objets recherchés.

D'après les concepteurs de Faster-RCNN, il est capable de répondre au besoin temps réel

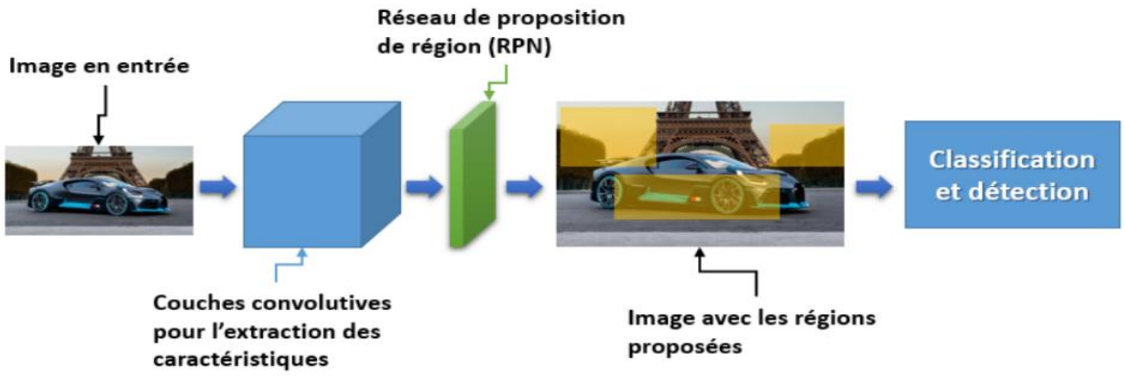


FIGURE II.5 – Schéma général de détection avec Faster-RCNN.

Singh et al [21]

En 2019 ont fait un travail pour la détection et la reconnaissance des caractères des plaques d'immatriculation Indiennes.

Démarche : La proposition des auteurs concernant la détection de la plaque d'immatriculation peut se résumer comme suit :

- Ils ont utilisé le modèle Faster-RCNN, un modèle doté d'un réseau de proposition de régions au sein de l'image, pour la détection de la plaque.
- Les régions proposées sont ensuite transformées en des régions d'intérêt qui ont la particularité d'avoir une dimension similaire, et cela, afin d'avoir des cartes de caractéristiques de taille identiques pour chaque région.
- Des techniques de classification et de régression de boites englobante sont utilisées pour raffiner la sélection de ces régions.

Utilisation de l'apprentissage avec transfert en utilisant des poids tirés d'un apprentissage antérieur avec le dataset COCO.

Dataset : Les auteurs ont utilisé un dataset contenant des images de plaques d'immatriculation Indiennes.

Il est constitué de 1000 images dont :

- 353 tirées de la plateforme Kaggle, et qui sont déjà labellisées.
- 647 tirées de Google Image, et qui ont été labellisées manuellement.

Résultats : Pour la détection de la plaque d'immatriculation, les auteurs ont seulement noté qu'ils ont atteint 99% en taux de précision pour 200 images de test.

Avantages :

- Utilisation de l'apprentissage avec transfert, ce qui s'est forcément avéré pratique vu le nombre très réduit de leur dataset.
- Utilisation du modèle Faster-RCNN, un modèle de détection d'objet réputé pour sa précision.

Désavantages :

- Ils auraient pu essayer d'appliquer leur méthode sur des datasets benchmarkés, et ce afin d'avoir une base de comparaison avec d'autres méthodes.
- Malgré le recourt à l'apprentissage avec transfert, l'entraînement du modèle FasterRCNN est connu pour être lent et gourmand en données d'entraînement.

II.3.3 CNNs en cascade [22]

Ce modèle a la particularité d'avoir plusieurs réseaux de neurones convolutifs disposés en séquence et qui sont de taille relativement petite. Le rôle de ces CNNs est simplement la proposition des régions candidates. D'après les concepteurs de ce modèle, il est censé augmenter la qualité de la précision tout en évitant le phénomène de sur-apprentissage.

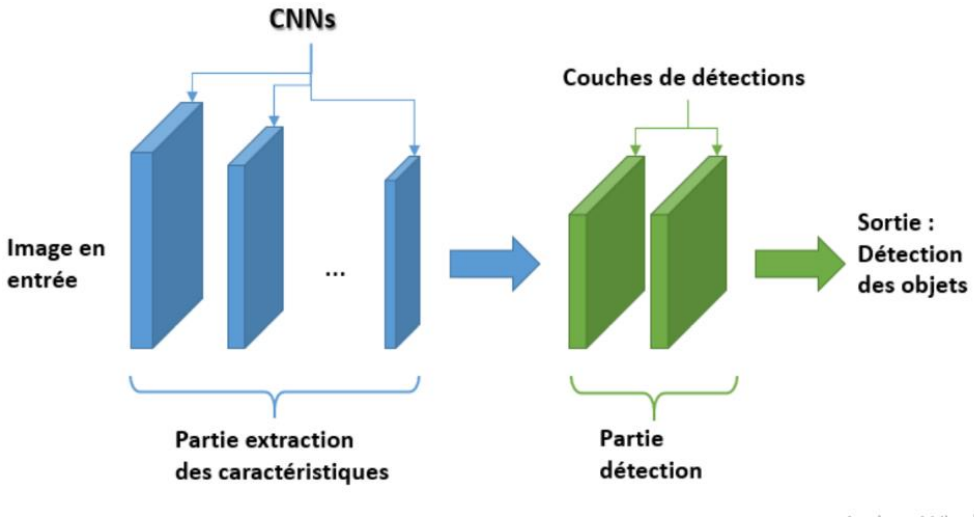


FIGURE II. 6 – Schéma général de détection avec les CNNs en cascade.

Wang et al en 2019 [23], sont Les auteurs d'un travail sur la détection des plaques d'immatriculation en utilisant ce modèlent contribué à la détection et à la reconnaissance des caractères de plaques d'immatriculation Chinoises.

Pour la partie détection, voici les contributions des auteurs :

- Ils ont proposé un modèle : MTLPR (Multi-Task License Plate Detection), qui est basée sur le fait de mettre en cascade des petits réseaux de neurones. Ils ont mis en cascade 3 réseaux de neurones : le P-Net, le R-Net, et le O-Net. Ces 3 réseaux sont suivi d'un CNN totalement connecté pour la partie classification.
- Les 3 réseaux de neurones ont pour objectif de générer des régions candidates pouvant contenir la plaque. Ils se basent sur des techniques de régression de la boite englobante et de NMS (Non-Maximal-Supression) afin de raffiner la sélection des régions.
- Ils ont proposé une augmentation des données d'entraînement basée sur : la géométrie des images, les couleurs, la luminosité, etc. Ils ont pu générer 500 000 images de plaque d'immatriculation.

Pour le Dataset : Il s'agit d'un dataset contenant 250 000 images de plaques d'immatriculation Chinoises. Il est divisé en 9 catégories, chacune présentant des caractéristiques différentes concernant : la luminosité, l'angle de la plaque, la distance, les conditions météorologiques, etc.

Les Résultats : Les auteurs ont comparé leur modèle (MTLPR) avec plusieurs autres modèles tirés de la littérature, et qui les ont, eux même, testés avec le même dataset (CCPD).

Modèles	FPS	Précision (%)	Rappel (%)
Faster-RCNN [20]	**15**	92.2	**88.3**
YOLO-V3 [20]	**42**	93.1	**93.1**
MTLPR	**65**	95.8	**96.9**
MTLPR + Augm.données	**65**	97.7	**97.3**

TABLE II.2 – Résultat de détection sur le dataset CCPD.
Augm. données : Augmentation des données d'apprentissage.
FPS : Frame par seconde, sous GPU Tesla K80. **Seuil IOU** : 0.7.

Le modèle proposé par les auteurs (MTLPR), combiné avec une augmentation des données d'apprentissage, présente les meilleurs résultats en terme de précision et de rappel. Il est aussi le meilleur en terme de FPS, il répond largement au besoin temps-réel.

Avantages :

- La méthode des 3 petits réseaux pour la détection de la plaque semble être très performante.
- Utilisation des techniques d'augmentation des données afin de rendre l'apprentissage du Modèle encore plus efficace.

Désavantages :

- Il n'y a pas de comparaison avec d'autres méthodes de d'autres auteurs.

II.4 Famille d'une seule phase

Cette famille, à l'inverse de l'autre, est caractérisée par le fait de pouvoir faire la détection des objets en une seule étape, et cela, à l'aide d'un unique réseau de neurones convolutif (CNN).

FIGURE II. 7 – Processus de détection en une seule phase.

Elle a la particularité de pouvoir faire un nombre fixe de prédictions sur une sorte de grille collée à l'image, à savoir, la prédiction des scores de confiances et les coordonnées des boites englobantes des objets recherchés. Cette famille est typiquement composée de modèles comme : YOLO, SSD, RetinaNet, etc.

La figure II.7 schématise le processus général de détection à l'aide de la famille à une seule phase.

Nous présentons par la suite quelques modèles connus de cette famille à travers un bref aperçu de leur principe de fonctionnement, le tout, accompagné de quelques travaux sur la détection des plaques d'immatriculation.

II.4.1 YOLOv1 [24]

Le modèle YOLO (You Only Look One) est un modèle de détection d'objets qui est très différents des modèles à proposition de régions. Il utilise un seul CNN pour la localisation des objets recherchés. Il connait un réel succès de fait de ses performances en terme de précision, mais surtout en terme de temps de détection.

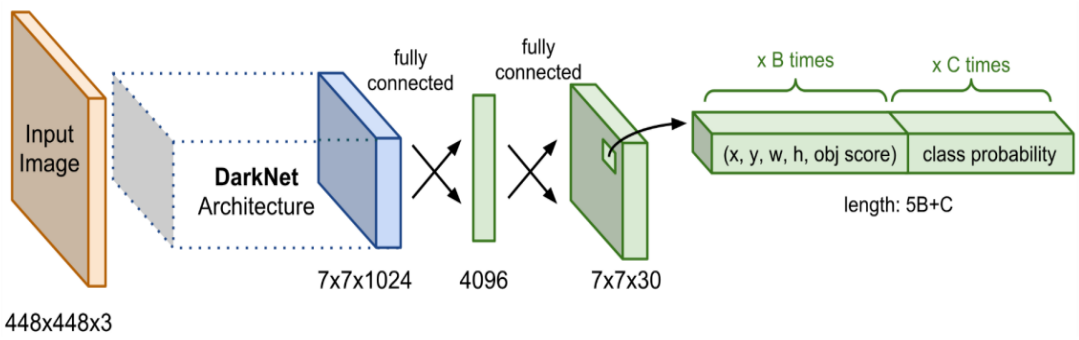


FIGURE II.7 – Architecture de YOLOv1. Cette figure a été tirée de ce travail [25].

Nous pouvons résumer son principe de fonctionnement comme suit :

1. À partir de l'image en entrée, YOLO divise l'image en grille de cellule de taille S × S.
2. Chaque cellule de la grille est associée à B ancres (boites englobantes) qui sont déjà prédéfinis.
3. Si un potentiel objet est détecté et que son centre se trouve sur une certaine cellule, cette dernière va être élue pour être la responsable de la prédiction de cet objet.
4. Chaque cellule élue va résulter une probabilité de classes ainsi que des coordonnées de la boite englobante.
5. En combinant ces résultats à travers un processus de régression, les coordonnées finales de la boite englobante de l'objet détecté vont être générées accompagnées d'un pourcentage de confiance.

Travaux

Voici un travail inspiré du modèle YOLO pour la détection des plaques d'immatriculation.

Xie et al. 2018 [29]

Dans ce travail les auteurs ont proposé une méthode pour la détection des plaques d'immatriculation Taiwanaises.

Démarche :

La méthode des auteurs se base sur le modèle YOLO (You Only Look One), sauf que par rapport au modèle original, ils ont apporté beaucoup de modification au but d'améliorer la précision de la détection des plaques. Les contributions des auteurs peuvent être résumées comme suit :

- Les auteurs ont appelé leur réseau de neurones MD-YOLO, référence au fait qu'il est basé sur le modèle YOLO.
- Ils se sont concentré sur le facteur de rotation de la plaque au sein de l'image. En effet, ils ont fait en sorte que la boite englobante générée puisse épouser les contours de la plaque pour donner, non pas à un simple rectangle, mais un parallélogramme aux formes de la plaque d'immatriculation.
- Pour la prédiction des angles de rotations des plaques, les auteurs ont proposé une version modifiée de la métrique IOU (Intersection Over Union).
- En plus du MD-YOLO qui inclue une phase de proposition des régions d'intérêt, ils ont ajouté un autre réseau CNN afin de proposer des régions d'attention. Ce CNN aura pour rôle de supprimer les informations redondantes et de proposer des régions dites d'attention, pour les expédier au MD-YOLO.
- Comme fonctions d'activation, ils ont utilisé la fonction d'identité et la fonction leaky, au lieu de RELU présente dans le YOLO original.
- Pour inclure le facteur angle de rotation des plaques au sein de l'image, ils ont proposé une fonction de perte afin de faciliter la détermination de ces angles.
- Les auteurs ont procédé à un apprentissage avec transfert en tirant les valeurs des poids d'un apprentissage sous le dataset ImageNet

Dataset

Les auteurs ont utilisé le dataset AOLP, initialement constitué de 3 catégories (AC,LE,RP). Les auteurs ont apporté deux changements :

- La catégorie RP a été modifié au niveau des labels, en effet, les auteurs ont ajouté le label a qui représente l'angle de rotation de la plaque.

- Les auteurs ont procédé à une technique d'augmentation des données suivant une stratégie spatiale basée sur la rotation des images. Cela permettra d'avoir plus de cas de rotation des plaques et ainsi pouvoir mieux entraîner le modèle.

Résultats

Le test de la méthode des auteurs a été réalisé sur le dataset AOLP. Les taux et temps de détection sont présentés dans les tableaux .. et ….. respectivement.

Méthodes	**AC**		**LE**		**RP**	
	P	**R**	**P**	**R**	**P**	**R**
Méthode des auteurs	99.51	99.51	99.43	99.43	99.46	99.46
Hsu et al.2012	91	96	91	95	91	94
Li et al. 2016	98.53	98.38	97.75	97.62	95.28	95.58

TABLE II.7 – Résultats comparatifs sur le dataset AOLP.
P : Précision(%) R : Rappel(%)

Méthodes	**CPU**	**GPU**
Méthode des auteurs	670 ms	7.7 ms
Hsu et al. 2012	90 ms	-
Li et al. 2016	-	2.5 s
SSD	-	50 ms
R-FCN	-	125 ms
Faster-rcnn	-	175 ms

TABLE II.8 – Temps de détection sur le dataset AOLP.
CPU : i7-2600
GPU : GTX980
s : seconde / ms : miliseconde

En terme de précision de détection, la méthode des auteurs donne des résultats extrêmement satisfaisants.

Concernant le temps de détection, la méthode des auteurs présente les meilleurs résultats sous une utilisation GPU. Sa confirme la possibilité d'un déploiement temps-réel.

Avantages

- Toutes les modifications et contributions des auteurs ont donné suite à un taux de détection excellent.
- Utilisation de l'apprentissage avec transfert.

Désavantages

- D'après les auteurs, il n'existe pas de travaux antérieurs basés sur la rotation des plaques, il reste donc à évaluer la méthode en la comparant avec des travaux futurs qui abordent la même optique.
- Le déploiement sous CPU aurait apporté d'avantage d'efficacité.

II.4.2 YOLOv2 [30]

YOLOv2 est la deuxième version de **YOLO**. Il présente une amélioration par rapport à la version une, mais conserve le même procédé général de détection. Il a été pensé pour l'amélioration du taux de précision, notamment, pour la détection des petits et grands objets au sein de l'image.
Par la suite nous avons présenté trois travaux qui abordent la détection des plaques d'immatriculation avec le modèle YOLOv2.

Laroca et al. 2018 [31] : Les auteurs de ce travail ont proposé une méthode pour la détection de la plaque d'immatriculation et la reconnaissance de ses caractères. Les plaques concernées sont de nature Brésiliennes.

Démarche

Pour la partie détection de la plaque, les auteurs ont utilisé les deux modèles YOLOv2 et Fast-YOLO sur deux dataset (SSIG et UFPR-ALPR). La méthode des auteurs peut être résumée comme suit :

- Les auteurs ont choisi de détecter d'abord le véhicule automobile pour qu'ensuite, à partir de ce dernier, détecter la plaque d'immatriculation.
- Les auteurs ont fait appel à un apprentissage avec transfert, en utilisant des poids pré-entraînés à partir du dataset ImageNet.
- Les auteurs ont utilisé les résultats de la reconnaissance des caractères afin de réduire le pourcentage des faux positifs des plaques détectées.

Dataset

Les auteurs ont utilisé deux datasets :

1. Le SSIG, un dataset contenant 2000 images de voitures Brésiliennes.

2. Le UFPR-ALPR, ce dataset est introduit par les auteurs durant ce même travail. Il contient 4500 images de voitures Brésiliennes prises sous différentes conditions d'éclairage et d'angles de vue, donnant ainsi une grande diversité au dataset.

Résultats

Pour la partie détection, les auteurs ont seulement relevé la précision et le rappel de leur méthode appliquée aux deux datasets.

-	Précision	Rappel	Temps détection
Détection des véhicules	99%	100%	4.0746 ms
Détection des plaques	100%	100%	4.0654 ms

TABLE II. – Résultat de détection sur la dataset SSIG.
Le nombre d'images de test est de 800 images.

Le seuil de confiance pour la détection des véhicules est fixé à 12.5%. C'est le modèle Fast-YOLO qui a été utilisé pour les deux détection. Le calcul du temps de détection a été fait avec une NVIDIA Titan XP.

-	Précision	Rappel	Temps détection
Détection des véhicules	99%	100%	11.1578 ms
Détection des plaques	-	98.33 %	3.9292 ms

TABLE II. – **Résultat de détection sur la dataset UFPR-ALPR.**
Le nombre d'images de test est de 1800 images.

Le seuil de confiance pour la détection des véhicules est fixé à 12.5%. C'est le modèle YOLOv2 qui a été utilisé pour la détection des véhicules. C'est le modèle Fast-YOLO qui a été utilisé pour la détection des plaques. La précision n'a pas été donnée pour la détection des plaques. Le calcul du temps de détection a été fait avec une NVIDIA Titan XP.

Un seuil de confiance faible a été utilisé afin d'augmenter les chances de détecter tous les véhicules.

Avantages

- Le résultant du processus de détection est largement suffisant pour faire du temps réel, et ce tout en présentant une performance remarquable.
- L'utilisation de l'apprentissage avec transfert.

Désavantages

- Puisque les objectifs des auteurs n'étaient pas que la détection de la plaque, ils ont préconisé l'utilisation d'un seuil de confiance relativement très faible, ce qui implique l'augmentation du risque d'avoir beaucoup de faux positifs. Une évaluation plus rigoureuse à propos de la détection aurait été plus parlante.

II.4.3 YOLOv3 [32]

YOLOv3 est la troisième version de YOLO dans, ils ont proposé cette architecture YOLOv3 pour localiser et reconnaitre les plaques d'immatriculation Bangla dans la ville m´métropolitaine de Dhaka, atteignant une précision de d´détection totale de plus de 95 Un autre travail a été fait sur les plaques d'immatriculation brésiliennes. Ils ont utilisé cette fois YOLOv3 en cascade et ils ont atteint une précision de 100% pour reconnaitre les voitures et leurs plaques d'immatriculation.

Cette architecture YOLOv3 introduit la faculté de détection à trois niveaux, c'est à dire, qu'il génère trois cartes de caractéristiques pour l'image d'entrée, et cela, afin de les fusionner et d'avoir une précision de détection d'avantages meilleure que celle de YOLOv2.

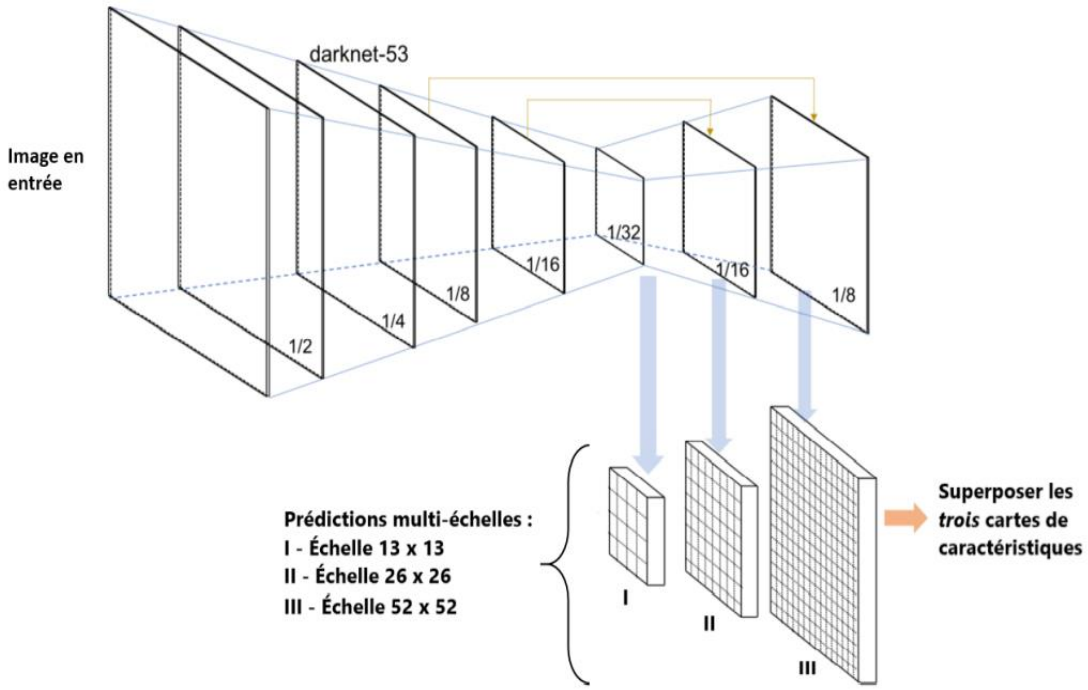


FIGURE II.8 – Architecture de YOLOv3.
La figure a été tiré de ce site [33].

Voici un travail s'inspirant de YOLOV3 pour la résolution du problème de détection des plaques d'immatriculation.

Peng et al. 2019 [34]

L'objectif des auteurs à travers ce travail est la détection des plaques d'immatriculation Chinoises.

Démarche

Voici ce que les auteurs ont proposé

- Ils ont proposé un modèle : ES-YOLOv3-Tiny, qui est inspiré du modèle YOLOv3 (YOLO version 3).
- Le modèle proposé peut détecter les plaques d'immatriculation ainsi que les voitures.
- Leur modèle peut effectuer deux détections. Si dans le cas où la première n'a pas pu identifier avec succès les plaques, un certain traitement va être effectué sur les images en sortie de la première détection avant de les repasser pour une deuxième détection.

- Le traitement utilisé après la première détection consiste en la proposition d'une zone d'attention au sein de l'image, et cela, afin de réduire le champ de détection pour la deuxième passe (la deuxième détection).

Dataset

Les auteurs ont récolté des images de plaque d'immatriculation Chinoises.

Il est divisé en deux catégories :

1. Péage à baïonnette : constitué de 20 000 images.
2. Parking de garage : constitué de 30 000 images.

Résultats

Les auteurs ont comparé leur modèle ES-YOLO-Tiny avec plusieurs autres modèles tirés de la littérature.

Modèle	Détection (%)	Tmps Détection (s)
RetinaNet	91.4	3.13
SSD	87.9	0.06
YOLOv3	92.3	0.04
YOLOv3-Tiny	89.9	0.006
ES-YOLOv3-Tiny	93.6	0.0077

TABLE II…… – Résultat de détection.

Tous les modèles du tableau sont testés avec le même dataset présenté ci-dessus. Le temps de détection est calculé sous GPU NVIDIA GTX 1060.

Le modèle des auteurs présente le meilleur taux de détection, et se classe 2ème pour le temps de détection.

Avantages

- Les auteurs se sont inspiré du modèle YOLO, un modèle présentant un très bon compromis taux/temps de détection.
- Principe d'une deuxième détection avec traitement, afin d'augmenter le taux de détection.

Désavantages

- Des tests sur des datasets benchmarkés auraient été mieux, pour qu'ensuite, pouvoir comparer la méthode des auteurs avec d'autres.

II.4.4 RetinaNet

Malgré leur vitesse ´élevée et leur simplicité, les détecteurs mono-phase ont trainé l'exactitude des détecteurs `a deux phases pendant des années. Lin et al. Ont découvert les raisons et proposé RetinaNet en 2018 [35]. Les auteurs proposent une fonction de perte, appelée perte focale, qui peut atténuer la perte attribuée à des exemples bien classés ou faciles. La perte focale se concentre sur les

exemples d'entrainement difficiles et évite le grand nombre d'exemples négatifs faciles écrasant le d´détecteur pendant l'entrainement.

RetinaNet est un réseau unifié composé d'un réseau backbone et de deux sous réseaux. Le backbone (Feature Pyramid Network (FPN) [ref]) est utilisé au-dessus d'une architecture ResNet et est responsable du calcul d'une carte de descripteur convolutives sur une image d'entrée entière et est un réseau convolutif autonome.

Le premier sous-réseau effectue la classification des objets convolutionnels sur la sortie du réseau principal ; le deuxième sous-réseau effectue une régression du cadre englobant convolutionnel. Les deux sous-réseaux présentent une conception simple qui a été proposé spécifiquement pour une détection dense mono-phase figure II.8

Les expériences ont montré que RetinaNet avec le réseau backbone ResNet-101-FPN obtenait 39,1% d'AP par rapport au DSSD513 de 33,2% d'AP sur l'ensemble de données MS COCO test-dev. RetinaNet a amélioré la précision de d´détection des objets petits et moyens par une grande marge.

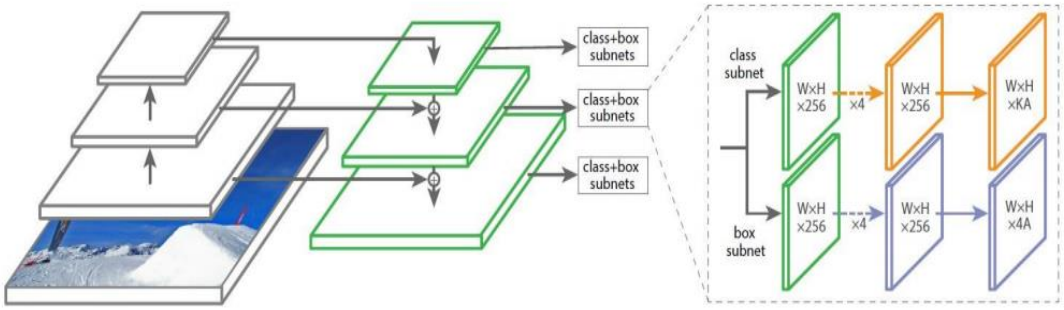


FIGURE II. 8 – Architecture de RetinaNet.
La figure a été tiré de ce travail [36]

L'algorithme RetinaNet et le réseau neuronal à convolution (CNN) ResNet-50 ont été utilisés par Safie et al. [ref] pour localiser et détecter les plaques d'immatriculation prises à partir d'images vidéo dans une application de reconnaissance automatique de plaque d'immatriculation. Les résultats ont montré une précision de 97,82%, ce qui est une amélioration par rapport à un classificateur à projection verticale et en cascade implémenté sur cette application. Des comparaisons sont faites entre une méthode proposée en basée sur YOLOv3 et certaines méthodes de détection d'objets, parmi lesquelles nous affinons RetinaNet. Les expériences ont montré que RetinaNet a échoué au moment du calcul, mais il a obtenu une précision considérable de 92%.

Avantages

- Utilisation de l'apprentissage avec transfert, cela a permis de ne pas entraîner le réseau Resnet-50, et donc, bénéficier d'un temps d'apprentissage assez réduit.
- RetinaNet démontre sa supériorité face à une méthode conventionnelle pour la détection des plaques d'immatriculation.

Désavantages

- Il aurait été mieux d'appliquer la méthode des auteurs sur des datasets benchmarkés afin d'avoir une meilleure idée des performances que peut donner RetinaNet.

II.5 Comparaison et synthèse des deux familles

Comparaison

À travers de ce qui a été présenté sur les deux familles des modèles de détection d'objets, nous pouvons tirer une comparaison sur leurs forces et leurs faiblesses, et cela, afin d'avoir une vision globale sur leur capacité, mais surtout, sur leur contexte d'utilisation. Le tableau II….. recense une synthèse de comparaison entre ces deux familles.

Famille	Forces	Faiblesses
Deux phases	• Du fait de leur étape de proposition des régions qui est à part, ils bénéficient d'une précision en détection relativement élevée. • Cette séparation des étapes (proposition des régions, classification) permet de mieux gérer le pipeline de détection, entre autres, essayer d'améliorer et d'optimiser les étapes de façon indépendante.	• La séparation des étapes amène à une consommation d'un temps d'exécution assez important, en effet, il très difficile de satisfaire le besoin temps réel. • Le temps d'apprentissage des modèles de cette famille est connu pour être assez gourmand, notamment, en terme de temps d'apprentissage mais aussi, en terme de données d'entraînement.
Une phases	• Exécuter la détection en une seule et grande étape a permis d'atteindre des temps de détection très satisfaisantes, ce qui place cette famille de modèles les leaders pour les applications temps réel. • Un temps d'apprentissage relativement court par rapport à la famille de deux phases.	• Le compromis du gain de temps de détection se ressens au niveaux de la précision. En effet, les modèles de cette famille ont tendance a être moins précis pour la détection des très petits ou très grands objets au sein de l'image. • Il est plus délicat d'essayer d'optimiser les différents composants de ces modèles de cette famille que ceux de la famille de deux phases.

	• Il existe même des architectures de modèle qui sont destinées pour des systèmes embarqués afin de satisfaire le besoin temps réel, sachant que, ces systèmes sont dotés de moyens de calcul très modestes.	

TABLE II. – Synthèse de l'état de l'art.

Synthèse

En vue de ce qui a été présenté ci-dessus, nous pouvons tirer le fait que la famille des modèles de détections d'objets à une seule phase peut être considérée comme meilleure que la famille à deux phases. En effet, malgré que cette dernière détient le titre de la plus précise en terme de détection, cela n'empêche pas l'autre (famille à une seule phase) de rivaliser avec elle, et ce en atteignant des taux de précision presque aussi semblables. En prime, il est à noter que la famille à une seule phase a la capacité de pouvoir gérer le besoin temps réel, capacité quasi-inexistante chez la famille à deux phases.

II.6 Défis de la détection des plaques d'immatriculation

Les algorithmes de détection doivent être capable de localiser les plaques d'immatriculation avec précision, en temps réel et de les convertir en numéro. Cependant, ce processus est susceptible d'être affecté par de nombreuses variations :

- La non uniformité des modèles de plaques d'immatriculation
- La faible résolution des plaques d'immatriculation
- Les conditions environnementales : luminosité, brouillard...etc.

II.7 Étude comparative

Étant donné de ce qui a été présenté dans la section précédente, cette section abordera une implémentation de deux modèles de la famille à une seule phase. Le but ici, sera de constater les performances de cette famille, et ce à travers quelques discussions sur des résultats obtenus sur des données réelles. Nous nous sommes donc proposés d'implémenter les deux modèles YOLOv3 et RetinaNet afin de réaliser la détection des plaques d'immatriculation des camions à la rentrer des CETs.

II.8 Conclusion

À travers ce chapitre, nous avons tenté d'exposer les différentes méthodes de Deep learning utilisées pour la résolution du problème de détection des plaques d'immatriculation. Il en ressort deux grandes familles, chacune, dotée de modèles de détection d'objets présentant des performances au summum de ce qui peut être fait dans la littérature. Cependant, il est bien de savoir qu'il s'agit de l'une de ces deux familles qui domine le devant de la scène. En effet, la famille à une phase présente les meilleurs résultats de détection pour les plaques d'immatriculation, que ce soit en terme de précision ou en de temps de détection.

Dans cette optique, nous avons pu voir et constater les performances de cette famille avec deux modèles de détection d'objets : YOLOv3 et RetinaNet qui va être utiliser pour l'implémentation de notre système, sur des données réelles de plaques d'immatriculation des camions à la rentrer des CETs dans le chapitre suivant. Aussi, nous allons exposer une modeste contribution ayant pour but d'améliorer le taux de détection.

Chapitre III

Proposition d'une fusion pour l'amélioration de la précision

Nous tentons à travers ce chapitre d'exposer une proposition afin d'augmenter la précision et l'efficacité de la détection des plaques d'immatriculation.
L'idée est d'arriver à fusionner les résultats de détection de plusieurs détecteurs pour une même image, et cela, à travers une procédure de fusion de données

Ce chapitre contient les sections suivantes :

- Section III.2 : Cette section contient l'idée générale de la contribution ainsi que ses motivations et son contexte.
- Section III.3 : Elle contient l'explication en détail de la contribution proposée à travers ses démarches et ses procédés.
- Section III.4 : Cette sec1tion contient un exemple de test de la fusion de deux modèles de réseau de neurones pour la détection des plaques d'immatriculation Algériennes.
- Section III.5 : Une conclusion autour de notre contribution ainsi que des perspectives.

III.2 Motivation et proposition

Motivation

Comme il a été vu dans le chapitre II concernant l'état de l'art, il existe plusieurs méthodes de détection des plaques d'immatriculation, chacune, présentant ses avantages et ses compromis. En effet, nous pouvons trouver dans la littérature des modèles de détection d'objets qui présentent des performances à différents niveaux, il serait intéressant de combiner ces modèles afin d'avoir les meilleurs résultats possibles en terme de précision. Quelques exemples de scénario :

- Deux modèles de détection des plaques d'immatriculation, l'un est performant dans de bonnes conditions de luminosité, alors que l'autre, il l'est dans des conditions de basse luminosité. Fusionner leur résultat serait bénéfique pour pouvoir assurer un plus grand spectre de lumière pour la détection.
- Fusionner un modèle qui est performant pour la détection des petits objets (les plaques d'immatriculation qui semblent être loin) avec un autre qui l'est tout aussi mais qu'avec les grands objets (les plaques d'immatriculation qui semble être près) dans l'image. La fusion des de ces deux modèles serait un avantage certain pour faire face au problème de dimension de la plaque durant le processus de détection.

- Deux modèles, chacun, serait entraîné sur un dataset différent destiné pour un type de plaque d'immatriculation différent (Exemple : plaques Algériennes, et plaques Tunisiennes). Fusionner leur performance serait intéressant afin de pouvoir couvrir une plus grande gamme de plaques d'immatriculation.

Nous pensons donc que fusionner ces différents modèles, ou différentes méthodes de détection, serait très avantageux dans l'attente d'une augmentation significative de la précision de la détection.

Pour situer notre contribution par rapport aux différentes notions liées au domaine de la fusion de données, nous présentons trois grandes taxonomies couvrant la majeure partie de l'éventail des concepts et méthodes de ce domaine :

- **Early fusion** : cette fusion concerne la fusion des données en entrée, dans notre cas, il s'agit des images pouvant contenir des plaques d'immatriculation. Cette fusion n'est pas en adéquation avec l'objectif de notre contribution vu que cette dernière, ne concerne qu'une seule image en entrée d'où nous voulons extraire les meilleures détections possibles.

- **Intermediate fusion** : c'est la fusion au niveau caractéristique, elle indique de combiner plusieurs vecteurs de caractéristiques provenant de différents extracteurs tel que les réseaux de neurones convolutifs, et cela, afin d'avoir un seul et grand vecteur de caractéristiques. Ce dernier pourra ensuite être passer vers une étape de classification. Le principal inconvénient de cette fusion est qu'elle est assez difficile à réaliser, en effet, il se trouve que combiner des vecteurs de caractéristiques dans un contexte de détection d'objets n'est pas tâche facile à mettre en pratique.

- **Late fusion** : c'est la fusion au niveau décision, elle indique de combiner les sorties de plusieurs classifieurs. Cette fusion ne présente pas une difficulté d'application aussi ardue que la précédente.

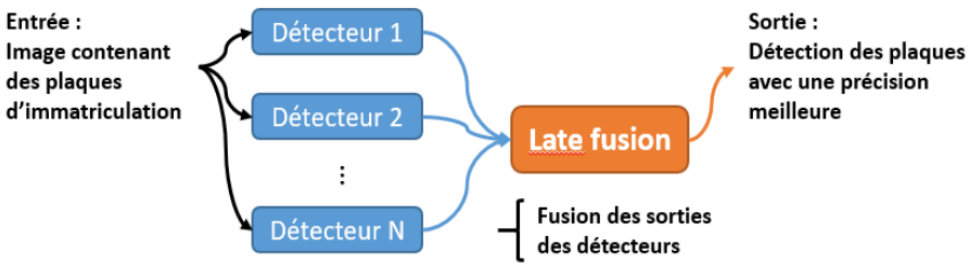


FIGURE III.1 – Late fusion pour la détection des plaques d'immatriculation. Chaque détecteur présente des détections pour la plaque d'immatriculation

Proposition

À la lumière des éléments présentés, et dans le cadre d'une fusion de type late fusion, nous allons proposer une méthode de fusion des résultats des différents détecteurs de plaque d'immatriculation, et ce afin d'augmenter la précision de la détection.

III.3 Fusion proposée

Afin de pourvoir fusionner les résultats des différents modèles de détection des plaques d'immatriculation, nous proposons plusieurs procédures qui seront capables de combiner les différentes sorties de ces détecteurs, et ainsi, générer de nouvelles détections encore plus précises.

Rappel : Caractéristiques d'un détecteur

Un détecteur a la particularité de pouvoir détecter les objets (pour notre cas : les plaques d'immatriculation) sur lesquels il a été entraînés. Il peut être caractérisé par plusieurs métriques d'évaluation, notamment, les pourcentages de faux positifs **(FPs)** et de faux négatifs **(FNs).**

Un détecteur de plaques d'immatriculation donne plusieurs détections en sortie, chacune de ces détections est caractérisée par :

• Une boite englobante : est un rectangle recouvrant la plaque détectée, et qui est localisée par deux point extrêmes, celui situé en haut à gauche, et celui en bas à droite. Chaque point est caractérisé par deux coordonnées x et y.

• Le pourcentage de confiance : généralement symbolisé par c, il représente le niveau de confiance accordé à la détection, c'est à dire, que nous pouvons être sûr à c% de l'exactitude de la détection. Inversement, si aucune détection n'est générée par le détecteur, cela veut dire que nous pouvons aussi être sûr à c% qu'il n'y a pas de plaque d'immatriculation.

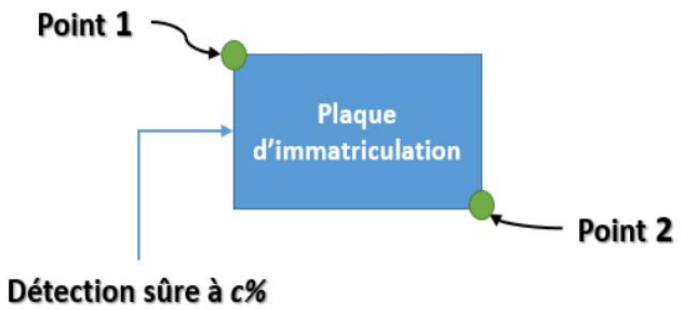


FIGURE III.2 – Sortie d'un détecteur de plaques d'immatriculation. Cette sortie représente une détection.

Hypothèse

Nous avons à notre disposition deux ou plusieurs détecteurs de plaques d'immatriculation. Pour des raisons de **simplicité**, nous avons décidé de combiner ces détecteurs deux à deux.

Ces raisons de simplicité viennent du fait que si nous voulions combiner plusieurs détecteurs, nous rencontrerions une difficulté très ardue pour le faire. En effet :

i - Chaque détecteur présente un certain nombre de détections de plaques d'immatriculation pour une image en entrée.

ii - Ces détections sont quasiment tous différentes puisque elles sont caractérisées par les coordonnées des boites englobantes et les pourcentages de confiance.

iii - Cela nous amène à dire que combiner ces détections revient à un problème combinatoire assez conséquent, et donc, assez difficile.

iv - C'est pour cela, d'ailleurs, que notre choix sur comment aborder la fusion a été porté sur la combinaison deux à deux des détecteurs.

Soit deux détecteurs de plaque d'immatriculation, chacun générant des détections au sein d'une même image. L'un génère N détections et l'autre M détections.

$$detector_1 = \{ td_1^1 , d^1_2 , ..., d_N{}^1 \} \quad (III.1a)$$

$$detector_2 = \{ td^2_1 , d^2_2 , ..., d_M{}^2 \} \quad (III.1b)$$

Toutes ces détections sont générées sous la contrainte que leurs pourcentages de confiance sont supérieurs à un certain seuil que nous l'appellerons ***seuil_confiance***.

$$\forall k \in \{1, ..., max(N, M)\}, \forall z \in \{1, 2\} \; ; \; c_k^z \geqslant \textbf{seuil\_confiance} \quad (III.2)$$

Avec : c^z_k est le pourcentage de confiance de la détection d^z_k .

À partir de ces détections, nous pouvons tirer deux cas de figure :

1. **Un même objet est détecté pas les deux détecteurs** : une détection d^1_i et une détection d^2_j font référencent au même objet (la plaque d'immatriculation) au sein de l'image. Le défi sera donc de pouvoir identifier *i* et *j* pour qu'ensuite, tenter de fusionner ces deux détections.

2. L'un des deux détecteurs détecte un objet et l'autre ne le détecte pas : le premier détecteur détecte une plaque d'immatriculation au sein de l'image, alors que le deuxième détecteur n'a pas pu détecter cette même plaque. Nous pouvons aussi avoir le scénario inverse.

Remarque : Le cas où aucun détecteur ne peut générer des détections, est d'ores et déjà à exclure puisque cela veut tout simplement dire qu'il y a une très forte probabilité qu'il n'existe aucune plaque d'immatriculation au sein de l'image.

III.3.1 Premier cas de figure

Dans un premier lieu, l'objectif sera de pouvoir identifier les détections qui font référence au même objet, et cela à partir des détections générées par les deux détecteurs (équation III.1).

En deuxième lieu, l'objectif sera de fusionner les détections similaires faisant référence au même objet.

a - Identification des détections similaires

Pour pouvoir faire l'identification des détections similaires entres les deux détecteurs, voici les étapes à suivre :

1. Faire un produit cartésien entre les détections du premier détecteur et le deuxième détecteur (entre III.1a et III.1b). Cela nous donnera des couples de détections comme suit :

$$COUPLES = (d_i^1, d_j^2)\ ,\ i = 1\_\_N,\ j = 1\_\_M \qquad \text{(III.3)}$$

2. Ensuite, appliquer la métrique **IOU (Intersection Over Union)** pour chaque couple. À chaque fois que la métrique IOU nous donnera une valeur supérieure à un certain seuil, nous considérerons que les deux détections du couple en question sont similaires.

Algorithme 2 Pseudo algorithme pour l'extraction des détections similaires

Entrée : Résultat du produit cartésien.
Sortie : Couples dont les détections sont similaires.

```
1: procédure GETDÉTECTIONSSIMILAIRES
2:     COUPLES = Résultat produit cartésien
3:     pour chaque couple ϵ COUPLES faire
4:         iou = getIOU(couple)
5:         si iou > seuil_IOU alors
6:             retourne couple.
7:         fin si
8:     fin pour
9: fin procédure
```

Un couple est représenté par (d^1_i , d^2_j).
La fonction *get*IOU retourne une valeur suivant la métrique **IOU**, vu dans la section II.2.2.

L'algorithme 2 retourne les couples de détections similaires en utilisant la métrique IOU.

b - Fusion des détections similaires

Après avoir identifié les détections similaires faisant référence au même objet (la plaque d'immatriculation), nous obtenons des couples de détections que nous allons essayer de les fusionner.

Pour exemple d'un couple de détection (d^1, d^2) tirée respectivement des deux détecteurs (III.1), chacune de ces deux détections est caractérisée par un pourcentage de confiance *c*, et une boite englobante localisée par deux point [(x_1, y_1);(x_2, y_2)]. Nous allons tenter de combiner ces deux détections afin de générer les nouvelles coordonnées de la nouvelle boite englobante ainsi que le nouveau pourcentage de confiance. Voici les étapes proposées :

1. Normaliser les deux pourcentages de confiance entre eux.

$$C_N^1 = \frac{C^1}{C^1+C^2} \qquad \text{(III.4a)}$$

$$C_N^2 = \frac{C^2}{C^1+C^2} \qquad \text{(III.4b)}$$

Avec : C_N^1 et C_N^2 sont respectivement les pourcentages de confiance normalisés des détections d^1 et d^2 .

2. Calculer les nouvelles coordonnées des deux points extrêmes (en haut à gauche et en bas à droite) de la nouvelle boite englobante.

$$\hat{x}_1^{12} = x_1^1 \times c_N^1 + x_1^2 \times c_N^2$$
$$\hat{y}_1^{12} = y_1^1 \times c_N^1 + y_1^2 \times c_N^2$$

$$\hat{x}_2^{12} = x_2^1 \times c_N^1 + x_2^2 \times c_N^2$$
$$\hat{y}_2^{12} = y_2^1 \times c_N^1 + y_2^2 \times c_N^2$$

Avec $[(\hat{x}_1^{12}, \hat{y}_1^{12}); (\hat{x}_2^{12}, \hat{y}_2^{12})]$ sont les coordonnées de la nouvelle boite englobante résultante de la fusion entre les deux boites englobantes des deux détections (d^1 et d^2).

3.Concernant le nouveau pourcentage de confiance cˆ 12, il va être tiré des deux pourcentages de confiance c^1 et c^2 . Nous avons choisi de faire la moyenne entre ces deux pourcentages.

$$\hat{c}^{12} = \frac{c^1 + c^2}{2}$$

Il faut savoir qu'il peut y avoir d'autres façons de faire pour tirer $\hat{c}_{12}$ tel que l'utilisation du *argmax* ou du *argmin* entre c^1 et c^2 .

III.3.2 Fusion deux à deux

La généralisation de la fusion se fera en combinant les résultats des détecteurs deux à deux.

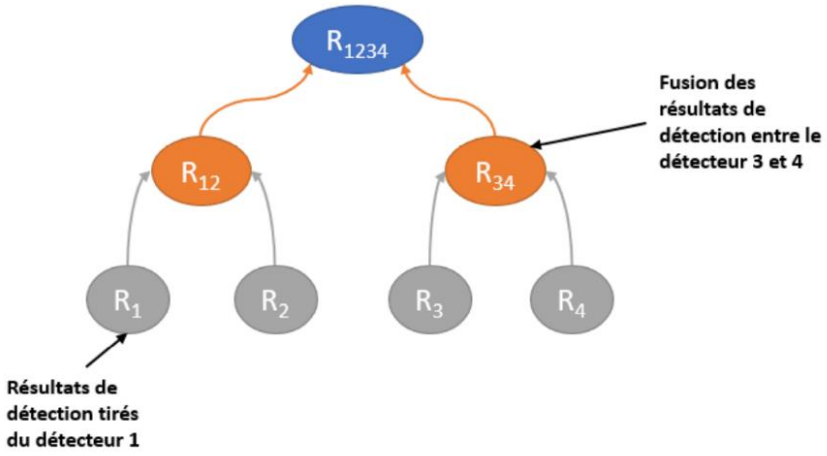


FIGURE III.3 – Fusion deux à deux.

Résultats de détection : signifie les détections des plaques d'immatriculation au sein de l'image

La figure III.3 présente le schéma général de fusion des résultats de plusieurs détecteurs.

Chaque deux résultats de détections (exemple R_1 et R_2) donnent naissance à un autre résultat (exemple R_{12}). Ce dernier est ensuite fusionné avec un autre et ainsi de suite jusqu'à avoir la fusion de l'intégralité des résultats des détecteurs.

Problème

La résultante d'une fusion est susceptible d'être utilisée pour une prochaine fusion, exemple : résultante R_{12} avec résultante R_{34}. Un problème se pose concernant la gestion du deuxième cas de figure durant le processus de fusion. En effet, il nous faudra générer les deux pourcentages de **faux positif (FP)** et de **faux négatif (FN)** de la résultante (R_{12} par exemple), et ce afin de pouvoir effectuer une fusion prochaine.

Solution

Comme solution au problème posé, nous avons proposé deux solutions différentes.

i - Solution Une :

L'idée est assez simple et elle consiste à générer le pourcentage de faux positif et de faux négatif de la résultante de la fusion, et ce à partir d'un dataset de test. Voici les étapes à suivre :

1. Un dataset de test contenant N images de plaques d'immatriculation labellisées.

2. Appliquer deux détecteurs sur ces N images afin de générer les détections des plaques. Nous aurons alors pour chaque images deux ensembles de détection, l'un du premier détecteur, et l'autre du deuxième détecteur.
3. Fusionner ces ensembles pour toutes les images à l'aide des processus de fusion proposés précédemment.
4. Calculer les nouveaux pourcentages de faux positif et de faux négatif de la résultante de la fusion. Ce processus signifierait que pour chaque nœud de la figure III.3, ce dernier détiendrait déjà les pourcentages de FP et le FN, et ce avant même de faire la fusion pour une image en entrée.

ii - Solution Deux :

Pour cette solution, nous avons décidé de générer les pourcentages de **FP** et de **FN** à partir des détecteurs inclus dans la fusion. Dans cette optique, nous nous sommes mis dans la situation où nous devons choisir entre les détections de detector1 et de detector2.

À partir de cette situation, ***4 différents cas*** peuvent être tirés en fonction des ***FPs*** et des ***FNs*** des deux détecteurs :

1. **Les détections de detector$_1$ sont les meilleures** :
   Cette situation veut dire que seulement les détections de detector1 sont laissées. Dans ce cas-là, FP^1 et FN^1 seront pris comme les nouveaux FP et FN de la résultante de la fusion.

2. **Les détections de detector$_2$ sont les meilleures** :
   Cette situation veut dire que seulement les détections de detector$_2$ sont laissées. Dans ce cas-là, FP^2 et FN^2 seront pris comme les nouveaux FP et FN de la résultante de la fusion.

3. **Les deux détecteurs sont considérés comme très bons** :
   Cette situation veut dire que toutes les détections des deux détecteurs sont laissées (sauf pour le cas où les détections similaires sont fusionnées). Dans ce cas-là, nous préconisons de baisser cette tendance à toujours prendre toutes les détections des deux détecteurs.

4. **Les deux détecteurs sont considérés comme mauvais** : Cette situation veut dire qu'aucune détection des deux détecteurs n'est gardées (sauf pour le cas des détections similaires où elles sont fusionnées). Les détections des deux détecteurs, prises individuellement, ont tendance à ne jamais être laissées. C'est pour cela d'ailleurs que nous préconisons de contrer cette tendance.

L'algorithme 4 présente la solution de chacun des cas motionné ci-dessus. L'entrée de l'algorithme consiste en les (FP^1 et FN^1) de **detector$_1$,** et les (FP^2 et FN^2) de **detector$_2$.**

L'objectif sera donc de générer **FP^R** et **FN^R** de la **résultante** de la fusion des deux détecteurs.

Algorithme 4 Génération du FP et du FN de la résultante de la fusion

Entrée : FP^1 et FN^1 , FP^2 et FN^2
Sortie : FP^R et FN^R ,
1: **procédure** GET FP^R FN^R
2**:** **si** ($FP^1 \leq FN^2$) and ($FN^1 < FP^2$) **alors** **1er cas**
3**:** $FP^R = FP^1$
4 : $FN^R = FN^1$
5 : **sinon si** ($FP^2 \leq FN^1$) and ($FN^2 \geq FP^1$) **alors** **2ème cas**
6: $FP^R = FP^2$
7: $FN^R = FN^2$
8: **sinon si** ($FP^1 \leq FN^2$) and ($FN^1 \geq FP^2$) **alors** **3ème cas**
9: FP^R = argmax (FP_1, FP_2)
10: FN^R = argmin (FN_1, FN_2)
11. **sinon si** ($FP^2 > FN^1$) and ($FN^2 < FP^1$) **alors** **4ème cas**
12. FP^R = argmin (FP_1, FP_2)
13: FN^R = argmax (FN_1, FN_2)
14 : **fin si**
15: **fin procédure**
FP^R et **FN^R** : sont respectivement les pourcentages de *faux positif* et de *faux négatif* de la résultante de la fusion entre *detector$_1$* et *detector$_2$*.

Résumé

Afin de fusionner les détections de plusieurs détecteurs, et pour des raisons de simplicité, nous prendrons deux à deux ces détecteurs pour faire fusionner leurs résultats (leurs détections des plaques d'immatriculation au sein d'une image).

Pour se faire, nous suivrons ces deux principales étapes :

1. Fusionner les détections des deux détecteurs suivant l'approche expliquée à travers les deux cas de figures présentés dans les sections III.3.1
2. Utiliser l'une des solutions présentées dans la section III.3.2 pour pouvoir générer les nouveaux pourcentages de faux positif (FP) et de faux négatif (FN) afin de les utiliser dans des fusions prochaines. À noter que pour savoir laquelle des solutions est la plus efficace, il faudrait faire des tests sur des données réelles.

CHIPITRE IV Implémentation

IV.1 Introduction

Après avoir présenté le domaine de la détection des plaques d'immatriculation, nous allons maintenant passer à la présentation de la problématique traitée dans ce rapport et les différents axes empruntés pour sa résolution. On commencera par l'énonciation des deux méthodes utilisées, suivie de la modélisation et la schématisation de chacune des deux expériences réalisées et enfin une discussion des résultats obtenus sera faite.

IV.2 Présentation des méthodes utilisées

Dans cette partie, nous allons présenter, après documentation, les deux méthodes que nous avons jugées être les meilleurs. Qui sont YOLOv3 et RetinaNet. Il est très difficile d'avoir une comparaison équitable entre les différentes architectures de détection d'objets. Il n'y a donc pas de réponse définitive sur quel modèle est le meilleur. Pour les applications réelles, nous faisons des choix pour équilibrer précision et vitesse.

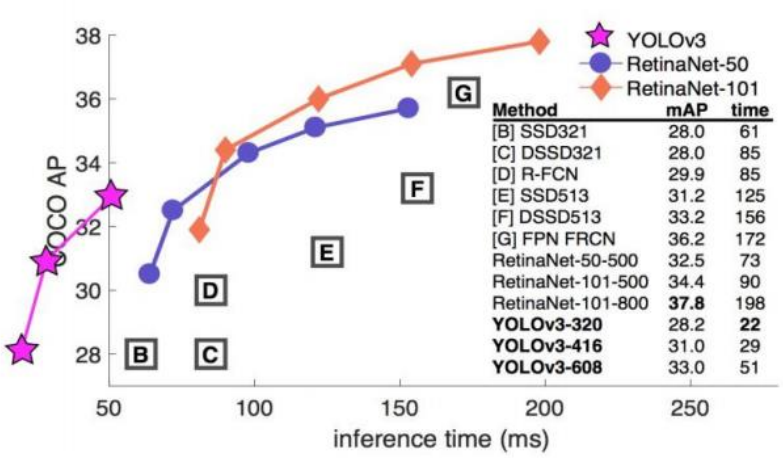


FIGURE IV.1 Les performances de YOLOv3 et RetinaNet dans COCO dataset **[37].**

La figure IV.1 montre que YOLOv3 est le meilleur en terme de vitesse par rapport aux autres architectures, avec une précision considérable et proche de la valeur maximale qui est atteinte par Ratina Net. C'est la raison pour laquelle on les a choisis pour être les parties participantes à la comparaison afin de voir leur comportement envers la détection de plaques d'immatriculation.

IV.2.1 You Only Look Once (YOLOv3)

Comme décrit dans le chapitre précédent, L'architecture de ce modèle est inspirée du modèle GoogLeNet pour la classification des images. Le réseau comprend 24 couches convolutives et 2 couches entièrement connectées. Les couches convolutives du réseau sont chargées d'extraire les descripteurs, tandis que les couches entièrement connectées prédisent les coordonnées et les probabilités de sortie des cadres englobants. La particularité de la troisième version de YOLO [37], YOLOv3 présente des performances élevés pour la détection des petits objets en gardant la précision et le rappel de la deuxième version.

Implémentations

Actuellement, il existe 3 implémentations principales de YOLOv3, chacune avec ses avantages et ses inconvénients

- Darknet [38] : Il s'agit de l'implémentation « officielle », créée par les mêmes personnes derrière l'algorithme. Il est écrit en C avec CUDA, il prend donc en charge le calcul GPU optimisé pour NVIDIA et Intel. Il s'agit en fait d'un Framework de réseau de neurones complet, il peut donc vraiment être utilisé pour d'autres objectifs en plus de la détection YOLO. L'inconvénient est que, puisqu'il est écrit à partir de zéro (non basé sur un Framework de réseau neurone stable comme CNN), il peut être plus difficile de trouver des réponses aux erreurs que vous pourriez rencontrer.

- AlexeyAB/darknet [39] : Cette implémentation est un fork de la version officielle de Darknet. C'est en fait une amélioration pour mieux supporter Windows et Linux. C'est l'implémentation nous avons utilisé. Le projet est hébergé sur la plateforme Github. C'est une excellente source pour trouver des conseils et des recommandations sur YOLO en général, comment préparer votre données d'entrainement, comment entrainer le réseau, comment améliorer la détection d'objet, etc.

- Darkflow [40] : Il s'agit d'une implémentation Python de Darknet en utilisant TensorFlow. Utiliser Darkflow uniquement sur CPU est plusieurs fois plus rapide que le Darkent d'origine. Le principal inconvénient est qu'il n'a pas été mis a` jour vers YOLOv3

Toutes ces implémentations sont « prêtes à l'emploi », ce qui signifie que vous devez fournir exactement les composants de l'environnement (en prenant en considération compatibilité entre les versions) dont elles ont besoin pour fonctionner. On peut commencer à faire des détections des images ou des vidéos immédiatement en utilisant des poids déjà entrainés disponibles en téléchargement pour faire les essais avant de passer à l'entrainement de votre propre dataset. Naturellement, cette détection sera limitée aux classes contenues dans les dataset utilisés pour obtenir ces poids. En effet, la véritable puissance de toutes ces implémentations est que vous pouvez les entrainer à détecter de nouvelles

classes. Comme la plupart des problèmes d'apprentissage automatique, l'un des premiers problèmes consiste à obtenir ou a créer notre propre dataset.

IV.3 Préparation du dataset

L'une des parties cruciales de la construction d'un système d'apprentissage automatique est la collecte d'un ensemble d'images de haute qualités.

IV.3.1 Collecte et Annotation des images

Pour mener nos expériences, nous avons choisi un dataset composé de photos prises sur des véhicules. Ce dataset contient 500 images de véhicules tirées d'une vidéo prise dans un carrefour routier.

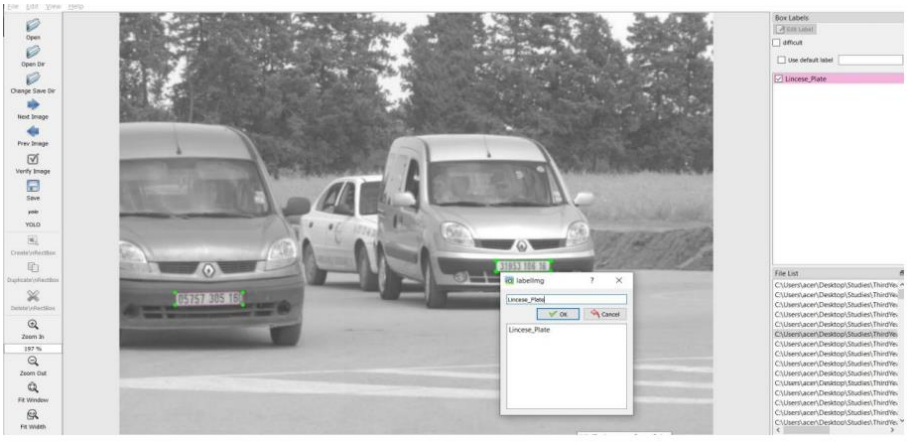
L'ensemble des images d'entrainement consiste en une série d'images, chacune doit venir avec un fichier texte indiquant les coordonnées et la classe de chacun des objets présents dans l'image (il est conseillé de rajouter si c'est possible quelques images avec aucun objet). C'est l'étape la plus pénible et la plus longue, toutes les images doivent être parcourues une par une et les objets doivent être annotés dans chaque image manuellement.

Il existe plusieurs outils qui peuvent être utilisés pour créer les annotations pour chacune des images qui feront partie de l'ensemble d'entrainement. Cela signifie, pour indiquer manuellement le « cadre entourant » contenant chacun des objets dans l'image et indiquer a` quelle classe appartient l'objet.

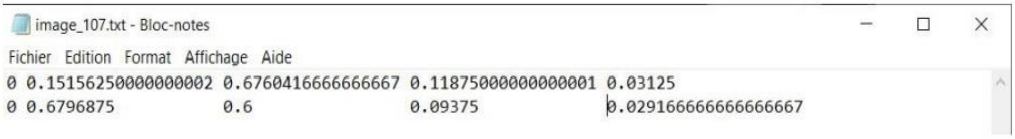
- Yolo Mark : Enregistre les annotations au format attendu par YOLO.

- LabelImg : Enregistre les annotations dans différents formats.

- BBox-Label-Tool. Enregistre les annotations dans un format qui n'est utilisé nativement par aucune des implémentations YOLO. En outre, la branche principale n'autorise pas plusieurs classes par image. Ce type de format, doit être converti au format YOLO.



Pour chaque image, nous allons obtenir un fichier correspondant contenant les annotations. Par exemple "image 107.txt" :



center x center y width height

- **object-id** représente le nombre correspondant à la catégorie d'objet (ici nous avons une seule classe.
- **center x** et **center y** représentent le point central du cadre de sélection. Mais ils sont normalisés pour varier entre 0 et 1 en divisant par la largeur et la hauteur de l'image.

  Par exemple, (0.25,0.75) est le point situé à 25% de la largeur et 75% de la hauteur. Nous pouvons multiplier ce nombre (0.25,0.75) par la largeur et la hauteur d'origine de l'image pour obtenir le point réel.

  Généralement, il est plus facile de prévoir des valeurs comprises entre 0 et 1 que des valeurs de coordonnées aléatoires.
- **width et height** représente la largeur et la hauteur du cadre de sélection. Encore une fois normalisé à la plage de 0 à 1 divisée par la largeur et la hauteur d'origine de l'image. Le fichier texte d'annotation généré pour chaque image contiendra chaque ligne comme ci-dessus et cela pour chaque cadre de l'image.

IV.3.2 Séparation de données apprentissage et test

Pour cette partie, on va suivre le découpage proposé par DarkNet. DarkNet attend un fichier texte répertoriant toutes les images qui seront utilisées pour la formation. Généralement, les gens utilisent 60 a` 90% de l'ensemble de données pour l'entrainement et gardent le reste pour les tests/la validation. Il n'y a pas vraiment de consensus ici sur les chiffres. Cela varie selon la situation.

Dans notre cas, nous avons un total de 300 images annotées, nous avons utilisé donc 250 images pour l'entrainement et 50 images pour la validation (avec un ration de 80% - 20%)

- **train.txt** : ce fichier contiendra la liste des images qui seront utilisées pour le train.
- **test.txt :** ce fichier contiendra la liste des images qui seront utilisées pour la validation.

IV.3.3 Création des fichiers nécessaires

Outre les annotations, il existe certains fichiers n´nécessaires lies aux données que DarkNet attend pour l'entrainement :

- classes.names : est le fichier ou` on liste les classes d'objets qu'on va détecter. Comme on ne s'entraîne que pour une seule classe, on ajoute simplement "Licence Plate EMP". Si on veut détecter plus d'un objet, on doit tous les ajouter dans ce fichier.

- obj.data : ce fichier contiendra les lignes suivantes :

  ˇ classes = 1 : c'est le nombre de classes d'objets qu'on apprend au réseau à détecter.

  ˇ train = /path to/train.txt

  ˇ test = /path to/test.txt

  ˇ names = /path to/classes.names

  ˇ backup = weights/ : c'est l'endroit ou` DarkNet enregistrera les poids issus de l'entrainement.

- yolov3.cfg : C'est le fichier de configuration principale de Yolo. Il contient les paramètres qui régissent le fonctionnement du réseau :

  ˇ nombre d'image par lot (batch) : c'est le nombre d'images introduit par itération

IV.4 Conception des expériences

Dans cette section on va s'étaler sur le corps principal de notre projet qui est la localisation des plaques d'immatriculation. Notre étude est la localisation des plaques d'immatriculation en suivant deux algorithmes notamment, YOLOv3 et RetinaNet. Cette étude est basée sur deux expériences, chacune est effectuée sur le même dataset. La première expérience adopte la localisation en utilisant l'architecture YOLOv3. La deuxième quant à elle adopte la localisation en utilisant l'architecture RetinaNet.

VI.4.1 la première expérience - YOLOv3

Il existe deux configurations de réseau possibles qui peuvent être utilisées pour entrainer, YOLOv3 ou tiny-YOLOv3. Comme son nom l'indique, tiny-YOLOv3 est un réseau plus petit, qui sera évidemment plus rapide à traiter mais qui souffrira d'une précision moindre.

Pour commencer l'entraînement, YOLOv3 nécessite un ensemble de poids convolutifs préentrainés sur Imagenet. Nous utilisons des poids du modéle darknet.

L'exécution se fera sur GOOGLE COLAB un outil de recherche d'apprentissage machine gratuit, convivial et facile à utiliser.

Les spécifications matérielles fournies dans l'environnement GOOGLE COLAB sont (source) :

- GPU : 1xTesla K80 , compute 3.7, having 2496 CUDA cores , 12GB GDDR5 VRAM

- CPU : 1xsingle core hyper threaded Xeon Processors @2.3Ghz i.e(1 core, 2 threads)

- RAM : 12.6 GB Available

- Disk : 33 GB Available

IV.5 Environnement de travail et outils utilisés

Voici la liste des technologies qui nous aidait pour réaliser les différentes expériences.

IV.5.1 Langage de programmation

Nous avons utilisé un langage très répondu actuellement, qui est le langage Python à cause de sa richesse en librairies.

Python : Python est un langage de programmation de haut niveau largement utilisé. Il a été initialement conçu par Guido van Rossum en 1991 et développé par Python Software Foundation. Il a été principalement développé pour mettre l'accent sur la lisibilité du code et de la syntaxe, il permet aux programmeurs d'exprimer des concepts avec moins de lignes de code.

Le langage python permet également de travailler rapidement et d'intégrer les systèmes plus efficacement. Il est relativement facile à apprendre et à transporter, ce qui signifie que ses instructions peuvent être interprétées sous plusieurs systèmes d'exploitation, notamment les systémes UNIX, Mac OS, MS-DOS, OS/2 et diverses versions de Microsoft Windows.

Interpréteur Pyhton 3.7.5 64x Un interpréteur est un programme qui lit et exécute du code. Cela inclut le code source, le code précompilé et les scripts. Pour le cas de Python, il vient avec un interpréteur qui fournit une fenêtre d'exécution de ligne commande efficace.

IV.5.2 OpenCV (4.2.1)

C'est une bibliothèque de fonctions de programmation principalement destinées à la vision par ordinateur en temps réel. Développé à l'origine par Intel, il a ensuite été soutenu par Willow Garage puis Itseez. La bibliothèque est multi-plateforme et gratuite pour une utilisation sous la licence open source BSD.

IV.5.3 Tensorflow gpu TensorFlow (2.1.0)

C'est une plateforme open source pour l'apprentissage automatique. Il dispose d'un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires qui permet aux chercheurs de faire évoluer l'état de l'art en Machine Learning et aux développeurs de créer et de

déployer facilement des applications propulsées par Machine Learning. La version GPU permet de faire des calculs sur GPU qui est plus rapide.

IV.5.4 DarkNet

C'est un framework de réseau de neurones open source écrit en C et CUDA. Il est rapide, facile à installer et prend en charge le calcul CPU et GPU.

IV.5.5 **Google Colab Colaboratory**, ou "Colab" en abrégé,

Est un service de virtualisation en ligne qui donne accès une instance d'un serveur Google. Il permet d'écrire et d'exécuter des scripts Python dans votre navigateur, avec :

- Aucune configuration requise.
- Accés gratuit aux GPU.
- Partage facile.

IV.6 Synthése des Résultats

Dans cette dernière partie, nous allons nous étaler sur les causes qui ont mené à l'obtention des résultats enregistrés.

IV.7 Conclusion

Dans ce chapitre nous avons réussi à définir convenablement le cadre de notre travail. On a commencé par la présentation du dataset utilisé et la façon dont il est exploité. Ensuite, nous avons illustré les deux approches implémentées.

Conclusion générale

Ce mémoire avait pour ambition d'étudier l'application du Deep learning sur la résolution du problème de détection des plaques d'immatriculation, et ce afin de répondre à la problématique autour du quel est le modèle de réseau de neurone qui est à même de satisfaire le besoin de détection avec les meilleures performances qui soient.

Il a fallu en premier lieu situer le problème à travers l'explication des différentes notions liées au problème de détection d'objets dans les images, ainsi que le Deep learning et les réseaux de neurones. Vient après, un état de l'art exposant les différentes méthodes basées sur les modèles de réseaux de neurones utilisées pour la détection des plaques d'immatriculation.

Au moyen de cet état de l'art, nous avons pu constater qu'il existait une famille de modèle de réseaux de neurones qui priment dans la détection des plaques d'immatriculation. En effet, la famille dite à une phase détient le meilleur compromis en terme de précision/temps de détection. Dans l'optique de vouloir développer un système ALPR, cette famille peut être considérée comme la plus adéquate. Effectivement, nous avons pu voir à travers l'implémentation sur des données réelles (plaques d'immatriculation Algériennes) de deux modèles (RetinaNet, YOLOv3) appartenant à cette famille que les résultats obtenus étaient très satisfaisantes dans le processus de détection.

Le bilan de ce qui a été motionné est que nous pouvons conclure que le modèle YOLOv3 parait comme le candidat idéal pour assurer la détection des plaques d'immatriculation, et ce de par sa précision mais surtout, de par son temps d'exécution. En effet, ce modèle pourra aisément répondre aux attentes du besoin temps réel dans les applications de type ALPR.

Dans l'initiative de contribuer à l'amélioration de la précision de la détection des plaques d'immatriculation, nous avons présenté une modeste contribution dans l'attente d'une réponse favorable à cette initiative. En effet, l'idée d'utiliser la fusion de type *late fusion* en fusionnant les résultats des modèles de détection des plaques semblait être une bonne idée, puisque la fusion des deux modèles (YOLOv3 et RetinaNet) a donné suite à une augmentation significative de la précision de la détection.

Ce travail de mémoire se voulait principalement tourner autour d'une recherche bibliographique sur l'impact de l'application du Deep learning pour la détection des plaques d'immatriculation, mais en vue de ce contexte, il était pertinent pour nous de contribuer à tirer parti de ce qui existe déjà, et ce dans le but d'implémenter notre application afin de détecter les plaques d'immatriculation des véhicules Algériennes.

Comme perspective pour ce travail, nous proposons de :

- Pousser encore plus nos expériences de fusion, notamment, l'utilisation d'un troisième modèle en plus de RetinaNet et de YOLOv3, et ce afin de faire face à d'éventuelles lacunes, sinon, apporter plus légitimité et de pertinence à notre contribution.

- Solliciter la fusion de type intermediate fusion dans un cadre de détection d'objet.

Bibliographie

[1] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with Deep learning : A review. IEEE transactions on neural networks and learning systems, 30(11) :3212–3232, 2019.

[2] A. Burkov. The Hundred-Page Machine Learning Book. Andriy Burkov, 2019.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams,"Learning internal representation by back-propagation of errors," Nature, vol. 323, no. 323, pp. 533–536, 1986.

[4] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath et al., "Deep neural networks for acoustic modeling in speech Recognition :The shared views of four research groups," IEEE Signal Process. Mag., vol. 29, no. 6, pp. 82–97, 2012.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei,"Imagenet : A large-scale hierarchical image database," in CVPR, 2009.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in NIPS, 2012.

[7] S. Ioffe and C. Szegedy, "Batch normalization : Accelerating deep network training by reducing internal covariate shift," in ICML, 2015.

[8] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat : Integrated recognition, localization and detection using convolutional networks," arXiv:1312.6229, 2013.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in CVPR, 2015.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv :1409.1556, 2014

[11] K. He, X. Zhang, S. Ren, and J. Sun,"Deep residual learning for image recognition," in Proceedings of the IEEE Conférence on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[12] Frank Rosenblatt. The perceptron : a probabilistic model for information storage and organization in the brain. Psychological review, 65(6) :386, 1958.

[13] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. Climate research, 30(1) :79–82, 2005.

[14] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A Tutorial on the cross-entropy method. Annals of operations research, 134(1) :19–67, 2005.

[15] Sebastian Ruder. An overview of gradient descent optimization algorithms. arXiv Preprint arXiv :1609.04747, 2016.

[16] Diederik P Kingma and Jimmy Ba. Adam : A méthode for stochastic optimization. arXiv preprint arXiv :1412.6980, 2014.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.

[18] E.S. Olivas, J.D.M. Guerrero, M. Martinez-Sober, and Magdalena-Benedito. Handbook of Research on Machine Learning Applications and Trends : Algorithms, Methods, and Techniques : Algorithms, Methods, and Techniques. Information Science Reference, 2009.

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587, 2014.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn : Towards realtime Object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99, 2015.

[21] Jaskirat Singh and Bharat Bhushan. Real time indian license plate detection using deep neural networks and optical character recognition using lstm tesseract. In 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), pages 347–352. IEEE, 2019.

[22] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn : Delving into high quality object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 6154–6162, 2018.

[23] Wanwei Wang, Jun Yang, Min Chen, and Peng Wang. A light cnn for end-to-end car license plates detection and recognition. IEEE Access, 7 :173875–173883, 2019.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once : Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 779–788, 2016.

[28] Lilian Weng. Object detection part 4 : Fast detection models. lilianweng.github.io/lil-log, 2018.

[29] Lele Xie, Tasweer Ahmad, Lianwen Jin, Yuliang Liu, and Sheng Zhang. A new cnn-based method for multi-directional car license plate detection. IEEE Transactions on Intelligent Transportation Systems, 19(2) :507–517, 2018.

[30] Joseph Redmon and Ali Farhadi. Yolo9000 : better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7263–7271, 2017.

[31] Rayson Laroca, Evair Severo, Luiz A Zanlorensi, Luiz S Oliveira, Gabriel Resende Gonçalves,

William Robson Schwartz, and David Menotti. A robust real-time automatic license plate recognition based on the yolo detector. In 2018 International Joint Conference on Neural Networks (IJCNN), pages 1–10. IEEE, 2018.

[32] Joseph Redmon and Ali Farhadi. Yolov3 : An incremental improvement. arXiv preprint arXiv :1804.02767, 2018.

[33] Hirota Honda. Reproducing training performance of yolov3 in pytorch (part1). www. medium.com, Avril 2020.

[34] Yongrong Peng, Hong Li, and Zheman Qian. A new end-to-end secondary network for high-efficient vehicles and license plates detection. In 2019 International Conference on Smart Grid and Electrical Automation (ICSGEA), pages 6–9. IEEE, 2019.

[35] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," IEEE transactions on pattern analysis and machine intelligence, 2018.

[36] Hwejin Jung, Bumsoo Kim, Inyeop Lee, Minhwan Yoo, Junhyun Lee, Sooyoun Ham, Okhee Woo, and Jaewoo Kang. Detection of masses in mammograms using a one-stage object detector based on a deep convolutional neural network. PLOS ONE, 13 :e0203355, 09 2018.

[37] J. Redmon, and A. Farhadi . "YOLOv3 : An Incremental Improvement." arXiv :1804.02767, 2018.

[38] https ://pjreddie.com/darknet 30-06-2020

[39] https ://github.com/AlexeyAB/darknet 30-06-2020

[40] https ://github.com/thtrieu/darkflow 30-06-2020